N° d'ordre

THESE

présentée à l'Université Scientifique et Médicale

de Grenoble

pour obtenir

le grade de Docteur de Troisième Cycle

"INFORMATIQUE"

par

Gérard TERRINE

CONSTRUCTION AUTOMATIQUE D'ANALYSEURS SYNTAXIQUES ASCENDANTS DETERMINISTES A PARTIR DE CF GRAMMAIRES EVENTUELLEMENT NON DE CONTEXTE BORNE

Thèse soutenue le 11 Mars 1972, devant la Commission d'examen :

Monsieur	N. GASTINEL	Président
Messieurs	J.J. LOECKX	Examinateur
	M. NIVAT	Examinateur
	M. GRIFFITHS	Examinateur

N° d'ordre

THESE

présentée à l'Université Scientifique et Médicale

de Grenoble

pour obtenir

le grade de Docteur de Troisième Cycle

"INFORMATIQUE"

par

Gérard TERRINE

CONSTRUCTION AUTOMATIQUE D'ANALYSEURS SYNTAXIQUES ASCENDANTS DETERMINISTES A PARTIR DE CF GRAMMAIRES EVENTUELLEMENT NON DE CONTEXTE BORNE

Thèse soutenue le 11 Mars 1972, devant la Commission d'examen :

Monsieur	N. GASTINEL	Président
Messieurs	J.J. LOECKX	Examinateur
	M. NIVAT	Examinateur
	M. GRIFFITHS	Examinateur

UNIV.
LEIDEN
BIBL.

UNIVERSITE SCIENTIFIQUE

ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du Solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Guy	Clinique rhumatologique et hydrologie
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures

MM.	GASTINEL Noël	Analyse numérique
	GERBER Robert	Mathématiques Pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques Pures
Mme	KOFLER Lucie	Botanique et Physiologie végétale
MM.	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques Appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques Pures
	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du Solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
M.	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	RADHAKRISHNA Pidatala	Thermodynamique

PROFESSEURS SANS CHAIRE

MM. AUBERT Guy	Physique
BEAUDOING André	Pédiatrie
BERTRANDIAS Jean-Paul	Mathématiques Appliquées
BIARES Jean-Pierre	Mécanique
BONNETAIN Lucien	Chimie minérale
Mme BONNIER Jane	Chimie générale
MM. CARLIER Georges	Biologie végétale
COHEN Joseph	Electrotechnique
COUMES André	Radioélectricité
DEPASSEL Roger	Mécanique des Fluides
DEPORTES Charles	Chimie minérale
DESRE Pierre	Métallurgie
DOLIQUE Jean-Michel	Physique des Plasmas
GAUTHIER Yves	Sciences biologiques
GEINDRE Michel	Electroradiologie
GIDON Paul	Géologie et Minéralogie
GLENAT René	Chimie organique
HACQUES Gérard	Calcul numérique
JANIN Bernard	Géographie
Mme KAHANE Josette	Physique
MM. MULLER Jean-Michel	Thérapeutique
PERRIAUX Jean-Jacques	Géologie et minéralogie
POULOUJADOFF Michel	Electrotechnique
REBECQ Jacques	Biologie (CUS)
REVOL Michel	Urologie
REYMOND Jean-Charles	Chirurgie générale
ROBERT André	Chimie papetière
SARRAZIN Roger	Anatomie et chirurgie
SARROT-REYNAULD Jean	Géologie
SIBILLE Robert	Construction Mécanique
SIROT Louis	Chirurgie générale
Mme SOUTIF Jeanne	Physique générale
M. VALENTIN Jacques	Physique nucléaire

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle AGNIUS-DELORD Claudine	Physique pharmaceutique
ALARY Josette	Chimie analytique
MM. AMBLARD Pierre	Dermatologie
AMBROISE-THOMAS Pierre	Parasitologie
ARMAND Yves	Chimie
BEGUIN Claude	Chimie organique
BELORIZKY Elie	Physique
BENZAKEN Claude	Mathématiques Appliquées
Mme BERTRANDIAS Françoise	Mathématiques Pures
MM. BLIMAN Samuel	Electronique (EIE)
BLOCH Daniel	Electrotechnique
Mme BOUCHE Liane	Mathématiques (CUS)
MM. BOUCHET Yves	Anatomie
BOUSSARD Jean-Claude	Mathématiques Appliquées
BOUVARD Maurice	Mécanique des Fluides
BRIERE Georges	Physique expérimentale
BRODEAU François	Mathématiques (IUT B)
BRUGEL Lucien	Energétique
BUISSON Roger	Physique
BUTEL Jean	Orthopédie
CHAMBAZ Edmond	Biochimie médicale
CHAMPETIER Jean	Anatomie et organogénèse

MM. CHIAVERINA Jean	Biologie appliquée (EFP)
CHIBON Pierre	Biologie animale
COHEN-ADDAD Jean-Pierre	Spectrométrie physique
COLOMB Maurice	Biochimie médicale
CONTE René	Physique
CROUZET Guy	Radiologie
DURAND Francis	Métallurgie
DUSSAUD René	Mathématiques (CUS)
Mme ETERRADOSSI Jacqueline	Physiologie
MM. FAURE Jacques	Médecine légale
GAVEND Michel	Pharmacologie
GENSAC Pierre	Botanique
GERMAIN Jean-Pierre	Mécanique
GIDON Maurice	Géologie
GRIFFITHS Michaël	Mathématiques Appliquées
GROULADE Joseph	Biochimie médicale
HOLLARD Daniel	Hématologie
HUGONOT Robert	Hygiène et Médecine préventive
IDELMAN Simon	Physiologie animale
IVANES Marcel	Electricité
JALBERT Pierre	Histologie
JOLY Jean-René	Mathématiques Pures
JOUBERT Jean-Claude	Physique du Solide
JULLIEN Pierre	Mathématiques Pures
KAHANE André	Physique générale
KUHN Gérard	Physique
Mme LAJZEROWICZ Jeannine	Physique
MM. LAJZEROWICZ Joseph	Physique
LANCIA Roland	Physique atomique
LE JUNTER Noël	Electronique
LEROY Philippe	Mathématiques
LOISEAUX Jean-Marie	Physique Nucléaire
LONGEQUEUE Jean-Pierre	Physique Nucléaire
LUU DUC Cuong	Chimie Organique
MACHE Régis	Physiologie végétale
MAGNIN Robert	Hygiène et Médecine préventive
MARECHAL Jean	Mécanique
MARTIN-BOUYER Michel	Chimie (CUS)
MAYNARD Roger	Physique du Solide
MICOUD Max	Maladies infectieuses
MOREAU René	Hydraulique (INP)
NEGRE Robert	Mécanique
PARAMELLE Bernard	Pneumologie
PECCOUD François	Analyse (IUT B)
PEFFEN René	Métallurgie
PELMONT Jean	Physiologie animale
PERRET Jean	Neurologie
PERRIN Louis	Pathologie expérimentale
PFISTER Jean-Claude	Physique du Solide
PHELIP Xavier	Rhumatologie
Mle PIERY Yvette	Biologie animale
MM. RACHAIL Michel	Médecine interne
RACINET Claude	Gynécologie et obstétrique
RICHARD Lucien	Botanique
Mme RINAUDO Marguerite	Chimie macromoléculaire
MM. ROMIER Guy	Mathématiques (IUT B)
ROUGEMONT (DE) Jacques	Neuro-Chirurgie
STIEGLITZ Paul	Anesthésiologie

	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du Solide
	PHELIP Xavier	Rhumatologie
Mle	PIERY Yvette	Biologie animale
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RICHARD Lucien	Botanique
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	ROUGEMONT (DE) Jacques	Neuro-chirurgie
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physiologie
Mme	RENAUDET Jacqueline	Microbiologie

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41534904_0029

Je tiens à remercier

Monsieur N. GASTINEL, *Professeur à l'Université Scientifique et Médicale de Grenoble, qui m'a fait l'honneur de présider le jury,*

Monsieur le Professeur J.J. LOECKX, *qui m'a accueilli pendant un an dans son équipe de recherche à l'Université d'ENSCHEDE (Pays-Bas) et qui a supervisé mes travaux,*

Monsieur M. NIVAT, *Maître de Conférences à l'Université de Paris IX, et Monsieur* J.C. BOUSSARD, *Maître de Conférences à l'Institut National Polytechnique de Grenoble, qui ont bien voulu participer au jury,*

Monsieur M. GRIFFITHS, *Maître de Conférences à l'Université Scientifique et Médicale de Grenoble, qui a été mon initiateur et ne m'a jamais ménagé son aide et ses encouragements amicaux,*

J. VAN DEN BROEK, *étudiant à l'Université d'Eindhoven (Pays-Bas), ainsi que* J. ENGELFRIED *et* J. VAN LOON, *assistants du Professeur LOECKX, pour les intéressantes discussions auxquelles ils se sont prêtées,*

Les autorités de l'Université d'Enschede qui ont permis mon séjour en Hollande,

et toutes les personnes ayant contribué à la réalisation de cette thèse.

TABLE DES MATIERES

Pages

INTRODUCTION *2*

CHAPITRE 1 : Quelques rappels sur les grammaires context-free *7*

CHAPITRE 2 : Ensemble de coordonnées associé à une CF grammaire G *13*

CHAPITRE 3 : Génération des chaînes de contexte *24*

CHAPITRE 4 : Recherche du déterminisme de l'analyseur *34*

CHAPITRE 5 : L'analyseur déterministe *45*

CHAPITRE 6 : Comparaison avec d'autres sous-ensembles des C-F grammaires *52*

CHAPITRE 7 : Une extension possible de la méthode COORD(m, k) *66*

BIBLIOGRAPHIE : *89*

ANNEXES : Répertoire des principales notations et terminologies *91*

INTRODUCTION

INTRODUCTION

Les méthodes d'analyse syntaxique pour les langages générés par des grammaires "context-free" ont été l'objet de très nombreux travaux. La préoccupation dominante a porté sur les méthodes répondant aux demandes suivantes

1 - l'analyse syntaxique est déterministe et fonctionne strictement de gauche à droite.

2 - le programme analyseur syntaxique (ou bien la table de décision) doit pouvoi être généré(e) automatiquement à partir de la grammaire du langage concerné.

3 - le programme analyseur syntaxique (ou bien le programme utilisant la table de décision) doit avoir une taille "raisonnable" et fonctionner "rapidement".

4 - Le sous-ensemble des grammaires context-free auquel une méthode donnée s'applique doit être assez "grand".

Ces demandes ne sont pas indépendantes les unes des autres : par exemple le caractère déterministe de l'analyseur syntaxique garantit qu'on n'explore pas, ou bien simultanément (consommation d'espace et de temps) ou bien séquentiellement (consommation de temps) les alternatives d'une situation non déterministe en cours d'analyse; le point 1 concourt donc à la réalisation du point 3 dans la mesure où les moyens employés pour obtenir le déterminisme ne consomment pas le temps et l'espace que celui-ci permettrait d'escompter. Par contre les points 1 et 3 d'une part et le point 4 d'autre part sont antagonistes car généralité et efficacité sont difficiles à concilier dans un algorithme. Enumérons quelques méthodes d'analyse syntaxique répondant aux demandes 1 et 2.
parmi les méthodes desdendantes :

LL(1) : Cette méthode satisfait bien la demande 3, mais pas très bien la demande 4 (ce qui nécessite des transformations sur les règles de la grammaire); cette méthode est appréciée des écrivains de compilateur car l'analyseur fournit les numéros des productions utilisées dans l'ordre "naturel" (i.e. descendant) et ce avec un "look-ahead" d'un seul symbole terminal, permettant à certaines fonctions sémantiques d'être exécutées simultanément. ([GRI 69], [KUN 67], [BOR 71]).

parmi les méthodes ascendantes :

précédence : cette méthode satisfait la demande 3 mais pas très bien la demande 4 (d'où également la nécessité de transformer les règles de la grammaire); elle est souvent employée dans les compilateurs. ([FLO 63], [WIR 66]).

précédence totale et contexte borné (BRC(m, k)) : ces méthodes satisfont passablement la demande 4 et moins bien la demande 3. ([COL 67], [FLO 64], [EIC 64], [LOE 70]).

LR(k) : cette méthode satisfait très bien la demande 4 mais pas du tout la demande 3 en ce sens que les tables de décision générées ont une taille inacceptable; en pratique on s'efforce de se limiter à la valeur 1 pour le paramètre k ([KNU 65], [COU 68], KOR [69]).

Les appréciations portées sur ces quelques méthodes ont évidemment un caractère quelque peu vague dû à l'absence de critères précis d'évaluation.

La méthode d'analyse syntaxique qui va être décrite ici est une méthode ascendante répondant aux demandes 1 et 2 et s'efforçant d'être un compromis "équilibré" entre les demandes 3 et 4.

Cette méthode (ainsi que le sous-ensemble des grammaires context-free qu'elle accepte) est nommée COORD(m, k); en effet, l'idée essentielle est de définir un automate d'analyse syntaxique dont l'alphabet de pile est composé d'éléments appelés "coordonnées" alors que, si on excepte la méthode LR(k) l'alphabet de pile utilisé usuellement est l'union de l'alphabet terminal et de l'alphabet auxiliaire de la grammaire concernée. Ce changement d'alphabet donne, on le verra, des propriétés intéressantes permettant de définir une méthode d'analyse syntaxique dont le "coût" est nettement moindre que celui de la méthode LR(k) au prix d'une perte de généralité peu sensible (l'ensemble des grammaires COORD(m, k) inclut strictement l'ensemble des grammaires BRC(m, k) et est strictement inclus dans l'ensemble des grammaires LR(k)).

L'idée de ce changement d'alphabet est déjà à la base de la méthode LR(k) et explique sa généralité [KNU 65]; elle apparaît encore plus clairement dans la notion de grammaire caractéristique introduite par J.J. LOECKX qui lui permet de construire un automate générateur de chaînes dont le vocabulaire de pile se compose de coordonnées, cet automate lui servant à construire automatiquement une table de décision pour un analyseur à contexte borné.(Θ) C'est donc essentiellement à partir des notions définies dans [LOE 70] que ce travail a été mené (*).

L'exposé s'articule comme suit : les chapitres 1 et 2 sont consacrés au rappel de notions de base ainsi qu'à certaines notations nécessaires pour la suite; le chapitre 3 expose rapidement les règles de fonctionnement d'un automate générateur de chaînes de contexte dérivant directement de celui décrit dans [LOE 70]; c'est dans le chapitre 4 qu'est décrit l'algorithme qui, à partir de l'ensemble des chaînes de contexte associées à une grammaire produit, quand c'est possible, une table de décision pour un analyseur COORD(m, k) dont la construction et le fonctionnement sont montrés dans le chapitre 5. Le chapitre 6 est consacré aux comparaisons entre l'ensemble de grammaires COORD(m, k) et les ensembles LL(k), BRC(m, k) et LR(k). Enfin le chapitre 7 expose quelques conjectures sur une généralisation possible de la méthode COORD(m, k),sujet que nous envisageons d'étudier ultérieurement.

Nous avons programmé l'ensemble des algorithmes décrits en PL/1 et quelques résultats sont montrés en annexe. Une équipe de l'I.R.I.A. (MM. KALFON, DELBREIL, BATTAREL) se proposant d'écrire un interpréteur APL sur C.I.I. 10070 utilise actuellement ce programme pour le traitement de la phase syntaxique. Nous avons dû laisser de côté un certain nombre de questions d'ordre théorique classiquement évoquées lorsqu'on définit une classe de grammaires acceptées par une méthode d'analyse syntaxique (voir par exemple les parties IV, V et VI du magistral article de KNUTH [KNU 65] pages 625 à 639) de même que de longs exposés sur la program-

(*) pendant un séjour d'un an en 1970 à l'Université de Technologie de TWENTE (ENSCHE DE, PAYS-BAS) dans le Département de Mathématiques Appliquées du Pr. VAN SPIEGEL, sous la direction du Pr. LOECKX.

(Θ) J.J. LOECKX en a d'ailleurs déduit l'idée d'un analyseur ascendant déterministe dont le vocabulaire de pile est formé de coordonnées; on verra, dans le chapitre 4, que ce type d'analyseur est un cas particulier de celui que nous proposons.

mation effective de notre algorithme afin de centrer l'exposé sur l'itinéraire suivi d'une sensation initiale à la phase précédant immédiatement l'écriture du programme.

CHAPITRE 1

QUELQUES RAPPELS SUR LES GRAMMAIRES CONTEXT-FREE

L'énoncé de définitions bien connues risquant d'être fastidieux, on se bornera ici à rappeler des conventions de notation.

Une CF grammaire est un quadruplet :

$$G = (N, T, R, Z)$$

où N est le vocabulaire auxiliaire;

et T est le vocabulaire terminal;

rappelons que $N \cap T = \emptyset$ et notons $V = N \cup T$ le vocabulaire;

R est l'ensemble des règles; il y a une règle par symbole auxiliaire $A \in N$:

$$A \to \varphi_1 \mid \varphi_2 \mid \ldots \mid \varphi_{p(A)}$$

comportant $p(A) \geq 1$ productions $\varphi_i \in V^*$ dont A est dit le sujet;

il y a Card(N) règles et p(G) productions pour une grammaire G.

Z est l'axiome de G.

Autant que possible et sauf indication contraire le symbolisme suivant sera utilisé :

A, B, C, ... pour les éléments de N;

a, b, c, ... pour les éléments de T où il est commode d'inclure éventuellement ε la chaîne vide (*)

$\alpha, \beta, \gamma, \ldots$ pour les éléments de V;

$\omega, \varphi, \psi, \ldots$ pour les éléments de V^*;

z, y, x, ... pour les éléments de T^*;

ces symboles pouvant être indicés si besoin est.

La relation binaire $\to$ qui a servi à écrire les règles de la CF grammaire G définit un sous-ensemble de $N \times V^*$; l'extension de cette relation pour définir un sous-ensemble de $V^* \times V^*$ se fait comme suit :

$$\varphi \to \psi \Leftrightarrow \varphi = \varphi_1 A \varphi_2,\ \psi = \varphi_1 \omega \varphi_2 \text{ et } A \to \omega;$$

La relation $\to$ n'a aucune propriété de symétrie, réflexivité ou transitivité; par fermeture transitive de la relation $\to$, on obtient une nouvelle relation binaire transitive $\overset{+}{\to}$ qui définit un sous-ensemble de $V^* \times V^*$ comme suit : $\varphi \overset{+}{\to} \psi \Leftrightarrow \varphi = \varphi_0$, $\psi = \varphi_n$, $n > 0$ et $\varphi_i \to \varphi_{i+1}$ pour

$$0 \leq i < n;$$

de même, par fermeture transitive réflexive de la relation $\to$, on obtient une nouvelle relation binaire réflexive et transitive $\overset{*}{\to}$ qui définit un sous-ensemble de $V^* \times V^*$ comme suit :

$$\varphi \overset{*}{\to} \psi \Leftrightarrow \varphi \overset{+}{\to} \psi \text{ ou } \varphi = \psi;$$

(*) On verra ultérieurement l'utilité de cette convention qui est incorrecte en ce sens que T^* n'est plus un monoïde libre mais reste valide parce que nous ne considérons que le sous-ensemble L(G) de T^* où la décomposition d'une chaîne sur T est unique dans la mesure où G n'est pas ambiguë.

On introduit de façon analogue les relations binaires $\xrightarrow{L}$ et $\xrightarrow{R}$ qui se définissent comme suit :

$\varphi \xrightarrow{L} \psi \Leftrightarrow \varphi = xA\varphi_2, \psi = x\omega\varphi_2$ et $A \to \omega$;

$\varphi \xrightarrow{R} \psi \Leftrightarrow \varphi = \varphi_1 Ay, \psi = \varphi_1 \omega y$ et $A \to \omega$;

et qui par fermeture transitive et fermeture réflexive donnent les relations $\xrightarrow{L+}$, $\xrightarrow{R+}$, $\xrightarrow{L*}$, $\xrightarrow{R*}$;

On peut ainsi définir un certain nombre d'ensembles attachés à une CF grammaire G :

$PH(G) = \{\omega \mid Z \xrightarrow{*} \omega\}$ formé des phrases de G;
$PHD(G) = \{\omega \mid Z \xrightarrow{L*} \omega\}$ formé des phrases descendantes de G;
$PHA(G) = \{\omega \mid Z \xrightarrow{R*} \omega\}$ formé des phrases ascendantes de G;

$L(G) = PH(G) \cap T^* = PHD(G) \cap T^* = PHA(G) \cap T^*$ formé des phrases terminales de G; cet ensemble est le langage engendré par G; à noter que :

$PHD(G) \subseteq PH(G)$ et $PHA(G) \subseteq PH(G)$;

On peut enfin définir des ensembles attachés à une CF grammaire G et à toute phrase $x \in L(G)$:

$ASD(x, G) = \{\omega_i \mid 0 \leq i \leq m, \omega_0 = x, \omega_m = Z$ et $\omega_{j+1} \xrightarrow{L} \omega_j$ pour $0 \leq j < m\}$; m étant fonction de x, G et $\xrightarrow{L}$; est un ensemble ordonné de description descendante pour la structure de x relativement à G (ou arbre syntaxique de x).

$ASA(x, G) = \{\omega_i \mid 0 \leq i \leq n, \omega_0 = x, \omega_n = Z$ et $\omega_{j+1} \xrightarrow{R} \omega_j$ pour $0 \leq j < n\}$; n étant fonction de x, G et $\xrightarrow{R}$; est un ensemble ordonné de description ascendante pour la structure de x relativement à G (ou arbre syntaxique de x).

Une CF grammaire G est non ambigüe si pour tout $x \in L(G)$ il n'existe qu'un seul ensemble ASD(x, G) (resp. ASA(x, G)); on ne s'intéressera ici qu'aux CF grammaires non ambigües.

Un analyseur syntaxique descendant (resp. ascendant) pour un langage L(G) engendré par une CF grammaire G non ambigüe est un algorithme qui, selon nos conventions, a pour donnée une chaîne de longueur finie $x \in T^*$, qui lit x strictement de gauche à droite en tentant de calculer simultanément les éléments de ASD(x, G) (resp. ASA(x, G)) et qui s'arrête après un nombre fini d'opérations pour une des deux raisons suivantes :

1) $x \in L(G)$ auquel cas le résultat est l'ensemble ASD(x, G) (resp. ASA(x, G));

2) $x \notin L(G)$
en indiquant si l'arrêt est dû à la cause 1 ou à la cause 2.

Les CF grammaires considérées ici devront, outre être non ambigües, répondre à la condition suivante :

$\forall A \in N$, $\exists \varphi, \Psi, x$ tels que

$Z \overset{*}{\rightarrow} \varphi A \Psi \overset{*}{\rightarrow} x$ et $A \overset{+}{\rightarrow} A$ n'est pas vérifié

Exemple 1

Soit $G_0 = (N_0, T_0, R_0, Z)$
$N_0 = \{Z, R, S, M, N\}$
$T_0 = \{<, >, ., +, -, d, \varepsilon\}$

$R_0 = \{ Z \rightarrow < R >$
$R \rightarrow S\ M\ .\ M$
$S \rightarrow + \mid - \mid \varepsilon$
$M \rightarrow N \mid \varepsilon$
$N \rightarrow Nd \mid d \}$

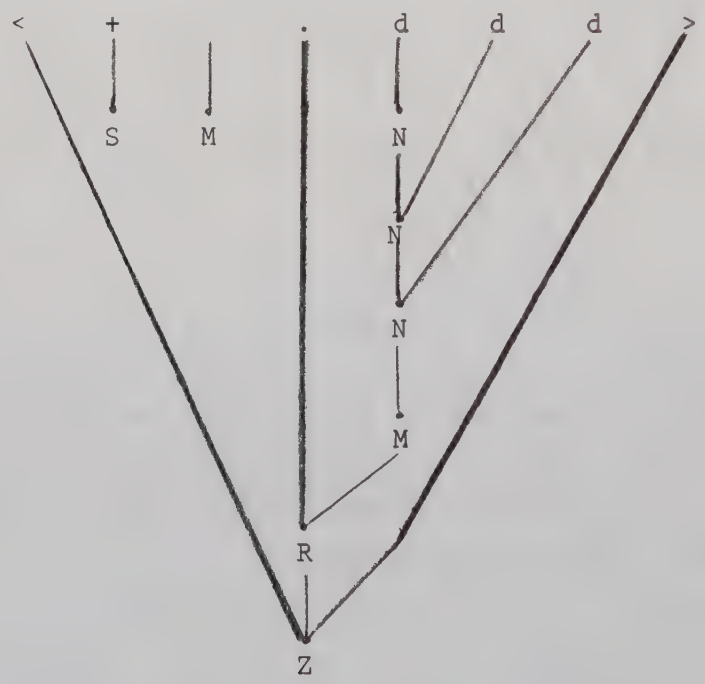
représentant la syntaxe des nombres écrits entre les marqueurs de début < et de fin > ayant un point décimal et éventuellement un signe, une suite de chiffres avant le point, une suite de chiffres après le point;
soit x = <+ . ddd> une phrase de $L(G_0)$;
on a $ASD(x, G_0)$ =

$\{ \omega_0 = <+ . ddd>$
$\omega_1 = <+ . Ndd>$
$\omega_2 = <+ . Nd>$
$\omega_3 = <+ . N>$
$\omega_4 = <+ . M>$
$\omega_5 = <+ M . M>$
$\omega_6 = <S M . M>$
$\omega_7 = <R>$
$\omega_8 = Z$

et $ASA(x,G_0)$ =

$$\{ \omega'_0 = \langle + \ . \ ddd \rangle$$
$$\omega'_1 = \langle S \ . \ ddd \rangle$$
$$\omega'_2 = \langle S \ M \ . \ ddd \rangle$$
$$\omega'_3 = \langle S \ M \ . \ Ndd \rangle$$
$$\omega'_4 = \langle S \ M \ . \ Nd \rangle$$
$$\omega'_5 = \langle S \ M \ . \ N \rangle$$
$$\omega'_6 = \langle S \ M \ . \ M \rangle$$
$$\omega'_7 = \langle R \rangle$$
$$\omega'_8 = Z \quad \}$$

et l'arbre syntaxique de x est l'arborescence construite à partir de l'un ou l'autre des ensembles $ASA(x, G_0)$ ou $ASD(x, G_0)$; cette arborescence comporte 8 sommets non pendants.



Enfin, suivant Knuth les notations suivantes seront utilisées (cf. [COU 68])

$k : \varphi$ = si $|\varphi| \geq k$ alors φ_1 tel que $\varphi_1\varphi_2 = \varphi$ et $|\varphi_1| = k$
sinon φ

$\varphi : k$ = si $|\varphi| \geq k$ alors φ_2 tel que $\varphi_1\varphi_2 = \varphi$ et $|\varphi_2| = k$
sinon φ

$H_k(\varphi) = \{k : x \mid \varphi \overset{*}{\geq} x\}$

Exemple 2 :

Soit une chaîne α = aAxy

on a :

3 : α = aAx

0 : α = ε (la chaîne vide).

4 : α = α

7 : α = α

$|\alpha|$: α = α

α : 2 = xy

α : 0 = ε

α : 4 = α

α : 6 = α

α : $|\alpha|$ = α

1 : (α : 3) = A = (2 : α) : 1

2 : (α : 3) = Ax = (3 : α) : 2

Soit la grammaire G_0 définie dans l'exemple 1;

$H_2(R)$ =

{+ ., + d, - ., - d, . >, d ., dd, . d ;}

$H_3(d)$ = {d};

D'autres notations seront éventuellement définies au moment de leur utilisation. Une dernière convention est que toute CF grammaire G envisagée devra avoir une règle initiale Z → ⊢ S ⊣ à une seule fin que toute phrase terminale soit encadrée par deux marqueurs, ⊢ et ⊣ où tous autres symboles.

CHAPITRE 2

ENSEMBLE DES COORDONNEES ASSOCIE A UNE CF GRAMMAIRE G

On va maintenant montrer qu'on peut définir un ensemble fini appelé C(G) à partir des règles d'une CF grammaire G par une simple opération de numérotation. Cet ensemble est muni d'une loi de concaténation qui permet de construire le monoïde $C^*(G)$. On définit alors naturellement un homomorphisme de $C^*(G)$ sur V^* (V étant le vocabulaire de G) et une grammaire GC déduite de façon unique de G telle que l'ensemble PH(G) soit l'image homomorphe de l'ensemble PH(GC). On verra que cette opération consiste fondamentalement à considérer que tout symbole(*) de V apparaissant dans une production des règles de la grammaire G doit se caractériser par son nom et par sa position d'occurrence. Ceci peut se faire en caractérisant la position d'occurrence par un couple d'indices (coordonnées) de l'ensemble C(G) et le nom par la connaissance de l'image homomorphe (dans V) de ce couple d'indices. L'exposé sera donc un "parallèle" pour $C^*(G)$ et GC de ce qu'on a vu dans le chapitre précédent.

A tout symbole(*) $\alpha \in V$ apparaissant dans les règles de G, on va faire correspondre un couple de nombres entiers $(i + k, j)$ de la façon suivante :
les règles étant écrites dans un ordre quelconque, sauf que la règle de l'axiome est la première, il en résulte que les productions correspondantes sont ordonnées et peuvent donc recevoir des indices g tels que $0 \leq g \leq p(G)$; de la même façon les symboles apparaissant dans les productions sont ordonnés de gauche à droite et peuvent donc recevoir des indices h tels que $1 \leq h \leq n(g)$, n(g) étant le nombre de symboles de la $g^{-ème}$ production; soit maintenant un symbole α quelconque intervenant dans les règles de G; ce peut être :

1) le sujet d'une règle; alors α est associé au couple de nombres $(i, 0)$ avec i = indice de la première production dont α est le sujet.

2) un symbole apparaissant dans une production; alors α est associé au couple de nombres $(i + k, j)$ avec

   i = indice de la première production de la règle où apparaît α;
   $i + k$ = indice de la production où apparaît α;
   j = indice de position de α dans la $i + k^{-ème}$ production;

(*) précisons : toute occurrence d'un symbole

Selon les besoins de la cause, on notera i + k. j ou g.j avec g = i + k un tel couple d'indices et si g.j est associé à une occurrence de $\alpha \in V$ dans les productions on dira que g et j en sont les coordonnées et appartiennent à l'ensemble C(G). Par contre un couple d'indices de la forme i.0 est associé à un symbole A de N sujet d'une règle de G et appartient à l'ensemble D(G).

En portant ces couples en position d'indices, la règle de sujet A d'une grammaire G s'écrit

$$A_{i.0} \to \alpha_{i.1}\alpha_{i.2} \cdots \alpha_{i.n(i)} \mid \alpha_{i+1.1}\ \alpha_{i+1.2} \cdots \alpha_{i+1.n(i+1)} \mid \cdots$$
$$\mid \alpha_{i+q.1}\alpha_{i+q.2} \cdots \alpha_{i+q.n(i+q)} \quad \text{avec } q = p(A) - 1$$

et i productions numérotées de 0 à i-1 ayant été écrites dans les règles qui précèdent la règle de sujet A.

On verra que les couples i + k.j avec $j \geq 1$ ont un rôle fondamental alors que les couples de forme i.0 représentent une facilité de notation.

En toute généralité, soit $A_j \in N$ le sujet de la $j^{ème}$ règle de G; on définit deux ensembles associés à G qui sont:

$$D(G) = \{i.0 \mid i = \sum_{j=1}^{\ell-1} p(A_j) \text{ pour } 0 \leq \ell \leq \text{Card}(N)\}.$$

Chaque sujet de règle étant associé d'une façon unique à un couple d'indices i.0 de l'ensemble D(G), nous pouvons, quand c'est utile, noter les éléments de N $A_{i.0}$ (il y a, rappelons-le, une règle par symbole auxiliaire);

et $C(G) = \{i + k.j \mid i.0 \in D(G),\ 0 \leq k < p(A_{i.0}),\ 1 \leq j \leq n(i + k)\}$,

qui est l'ensemble des coordonnées associé à une CF grammaire G.

Exemple 3

Soit la grammaire G_0 définie dans l'exemple 1; réécrivons-en les règles en ajoutant les indices

$$R_0 = \{Z_{0.0} \to <_{0+0.1}\ R_{0+0.2}\ >_{0+0.3}$$
$$R_{1.0} \to S_{1+0.1}\ M_{1+0.2}\ \cdot_{1+0.3}\ M_{1+0.4}$$
$$S_{2.0} \to +_{2+0.1} \mid -_{2+1.1} \mid 2+2.1$$

$M_{5.0}$ $N_{5+0.1}$ $5+1.1$

$N_{7.0}$ $N_{7+0.1}$ $d_{7+0.2}$ $d_{7+1.1}$

On a C(G) = {0.1, 0.2, 0.3, 1.1, 1.2, 1.3, 1.4, 2.1, 3.1, 4.1, 5.1, 6.1, 7.1, 7.2, 8.1}
les indices i + k.j étant écrits avec i + k effectué;
de plus :

n(0) = 3; n(1) = 4; n(2) = n(3) = n(4) = n(5) = 1;
n(7) = 2; n(8) = 1;

et on a D(G) = {0.0, 1.0, 2.0, 5.0, 7.0};

On peut munir l'ensemble C(G) de la loi de concaténation et construire ainsi le monoïde $C^*(G)$; on voit alors qu'il existe un homomorphisme de monoïdes de $C^*(G)$ sur W^* si on note W = V - {Z}; en effet soit une fonction symb définie comme suit :

1) $\forall i + k.j \in C(G)$, $symb[i + k.j] = \alpha$ avec $\alpha \in W$ et a les coordonnées i + k et j;

2) $\forall \overline{\varphi}$ et $\overline{\psi} \in C^*(G)$, $symb[\overline{\varphi}\,\overline{\psi}] = symb[\overline{\varphi}]\ symb[\overline{\psi}]$
en convenant d'utiliser la notation $\overline{\varphi}$, $\overline{\psi}$, ... pour les chaînes de $C^*(G)$. Cette fonction est, d'après sa définition même un homomorphisme de $C^*(G)$ sur W^* qui n'est injectif que dans le cas exceptionnel où tout symbole α de W n'a qu'une seule occurrence dans l'ensemble des productions de G.

L'homomorphisme symb permet de définir une relation d'équivalence R_{symb} sur $C^*(G)$ comme suit :

$$\overline{\varphi} \equiv \overline{\psi} \pmod{R_{symb}} \Leftrightarrow symb[\overline{\varphi}] = symb[\overline{\psi}]$$

On en déduit immédiatement que
$C^*(G)/R_{symb}$ et W^* sont isomorphes, et en particulier que $C(G)/R_{symb}$ et W sont en correspondance bijective; de plus les ensembles E(G) et N - {Z} sont en correspondance bijective, en notant E(G) = D(G) - {0.0}, donc :
$C(G)/R_{symb}$ est en correspondance bijective avec $E(G) \cup T$ et
$C^*(G)/R_{symb}$ est isomorphe à $\{E(G) \cup T\}^*$

ce qui permet de donner des noms aux classes d'équivalence ainsi définies.

Si un élément i + k.j de C(G) est tel que symb[i + k.j] = a élément de T, nous pourrons noter ceci par
i + k.j ∈ a i.e i + k.j appartient à la classe d'équivalence appelée a contenant tous les c + e.d tels que symb[c + e.d] = a.

Par contre, si un élément i + k.j de C(G) est tel que symb[i + k.j] = A élément de N nous noterons ceci par
i + k.j ∈ c.o où c.o ∈ E(G) est associé à l'occurrence unique de A comme sujet d'une règle; cette notation est donc équivalente à i + k.j ∈ A (qui a le même sens que ci-dessus mais elle sera plus pratique à utiliser.

A noter que, pour toute grammaire G ayant explicitement dans son vocabulaire terminal, la classe d'équivalence selon R_{symb} appelée ne contient que des couples de coordonnées du type i + k.1.

Exemple 4

Soit la grammaire G_o définie dans les exemples 1 et 3;
on a :

E(G) = {1.0, 2.0, 5.0, 7.0}
N - {Z} = {R, S, M, N}

qui sont évidemment en correspondance bijective;
les classes de $C(G)/R_{symb}$ sont :

1.0 = {0.2}; 2.0 = {1.1}; 5.0 = {1.2, 1.4};
7.0 = {5.1, 7.1}; < = {0.1}; > = {0.3};
. = {1.3}; + = {2.1}; - = {3.1};
d = {7.2, 8.1}; ε = {4.1, 6.1};

les classes 1.0, 2.0, 5.0, 7.0 peuvent être aussi bien nommées R, S, M, N;
on a W = {R, S, M, N, <, >, +, -, ., ε, d};

Nous allons maintenant définir des relations binaires $\twoheadrightarrow$, $\overset{+}{\twoheadrightarrow}$, $\overset{*}{\twoheadrightarrow}$ sur $C^*(G)$; définissons un sous-ensemble de E(G) X $C^*(G)$ par une relation binaire notée $\twoheadrightarrow$ comme suit :
On a : c + e.d $\twoheadrightarrow$ i + k.1 i + k.2 ... i + k.n(i + k) pour $\forall$ c + e.d ∈ C(G) tel que c + e.d ∈ i.0 et pour $\forall$ k tel que $0 \leq k < p(A_{i.0})$;
on procède à l'extension de cette relation pour définir un sous-ensemble de $C^*(G)$ X $C^*(G)$ comme suit :

$\overline{\varphi} \twoheadrightarrow \overline{\Psi} \Rightarrow \overline{\varphi} = \overline{\varphi}_1\ c + e.d\ \overline{\varphi}_2$, $\overline{\Psi} = \overline{\varphi}_1\ \overline{\omega}\ \overline{\varphi}_2$, et

$\exists i.0 \in E(G)$ tel que $c + e.d \in i.0$ et $\overline{\omega} = i + k.1\ i + k.2 \ldots i + k.n(i + k)$ pour quelque k tel que $0 \leqslant k < p(A_{i.0})$.

Il est évident que pour $\forall \overline{\varphi}$ et $\overline{\Psi}$ tels que $\overline{\varphi} \twoheadrightarrow \overline{\Psi}$, on a $\text{symb}[\overline{\varphi}] \rightarrow \text{symb}[\overline{\Psi}]$;

les relations $\overset{+}{\twoheadrightarrow}$ et $\overset{*}{\twoheadrightarrow}$ se déduisent par fermeture transitive et fermeture transitive réflexive de $\twoheadrightarrow$; il est non moins évident que :

pour $\overline{\varphi}$ et $\overline{\Psi}$ tels que $\overline{\varphi} \overset{+}{\twoheadrightarrow} \overline{\Psi}$ (resp. $\overline{\varphi} \overset{*}{\twoheadrightarrow} \overline{\Psi}$), on a $\text{symb}[\overline{\varphi}] \overset{+}{\rightarrow} \text{symb}[\overline{\Psi}]$ (resp. $\text{symb}[\overline{\varphi}] \overset{*}{\rightarrow} \text{symb}[\overline{\Psi}]$); de façon analogue les relations binaires $\overset{L}{\twoheadrightarrow}$ et $\overset{R}{\twoheadrightarrow}$ se définissent comme suit :

$\overline{\varphi} \overset{L}{\twoheadrightarrow} \overline{\Psi} \Leftrightarrow \overline{\varphi} = \overline{x}\ c + e.d\ \overline{\varphi}_2$, $\overline{\Psi} = \overline{x}\ \overline{\omega}\ \overline{\varphi}_2$,

$\text{symb}[\overline{x}] \in T^*$, $\text{symb}[c + e.d] \in N - \{Z\}$ et $c + e.d \twoheadrightarrow \overline{\omega}$;

$\overline{\varphi} \overset{R}{\twoheadrightarrow} \overline{\Psi} \Leftrightarrow \overline{\varphi} = \overline{\varphi}_1\ c + e.d\ \overline{y}$, $\overline{\Psi} = \overline{\varphi}_1\ \overline{\omega}\ \overline{y}$,

$\text{symb}[\overline{y}] \in T^*$, $\text{symb}[c + e.d] \in N - \{Z\}$ et $c + e.d \twoheadrightarrow \overline{\omega}$;

les relations $\overset{L+}{\twoheadrightarrow}$, $\overset{R+}{\twoheadrightarrow}$, $\overset{L*}{\twoheadrightarrow}$, $\overset{R*}{\twoheadrightarrow}$ s'en déduisent par fermeture transitive et fermeture réflexive des relations $\overset{L}{\twoheadrightarrow}$, $\overset{R}{\twoheadrightarrow}$; ainsi que précédemment il est évident que :

pour $\forall \overline{\varphi}$ et $\overline{\Psi}$ tels que $\overline{\varphi} \overset{L+}{\twoheadrightarrow} \overline{\Psi}$, on a $\text{symb}[\overline{\varphi}] \overset{L+}{\rightarrow} \text{symb}[\overline{\Psi}]$;

pour $\forall \overline{\varphi}$ et $\overline{\Psi}$ tels que $\overline{\varphi} \overset{R+}{\twoheadrightarrow} \overline{\Psi}$, on a $\text{symb}[\overline{\varphi}] \overset{R+}{\rightarrow} \text{symb}[\overline{\Psi}]$;

pour $\forall \overline{\varphi}$ et $\overline{\Psi}$ tels que $\overline{\varphi} \overset{L*}{\twoheadrightarrow} \overline{\Psi}$, on a $\text{symb}[\overline{\varphi}] \overset{L*}{\rightarrow} \text{symb}[\overline{\Psi}]$;

pour $\forall \overline{\varphi}$ et $\overline{\Psi}$ tels que $\overline{\varphi} \overset{R*}{\twoheadrightarrow} -$, on a $\text{symb}[\overline{\varphi}] \overset{R*}{\rightarrow} \text{symb}[\overline{\Psi}]$;

On peut définir les ensembles :

$\overline{PH}(G) = \{\overline{\omega} \mid 0.1\ 0.2\ 0.3 \overset{*}{\twoheadrightarrow} \overline{\omega}\}$;

$\overline{PHD}(G) = \{\overline{\omega} \mid 0.1\ 0.2\ 0.3 \overset{L*}{\twoheadrightarrow} \overline{\omega}\}$;

$\overline{PHA}(G) = \{\overline{\omega} \mid 0.1\ 0.2\ 0.3 \overset{R*}{\twoheadrightarrow} \overline{\omega}\}$;

et on a : $\text{symb}[\overline{PH}(G)] = PH(G)$;

$\text{symb}[\overline{PHD}(G)] = PHD(G)$;

$\text{symb}[\overline{PHA}(G)] = PHA(G)$;

continuant ainsi on peut définir :

$\overline{L}(G) = \overline{PH}(G) \cap \text{symb}^{-1}[T^*] = \overline{PHD}(G) \cap \text{symb}^{-1}[T^*]$

$= \overline{PHA}(G) \cap \text{symb}^{-1}[T^*]$;

et on a $\text{symb}[\overline{L}(G)] = L(G)$;

Soit une phrase terminale $x \in L(G)$; étant donné que G n'est pas ambiguë, il y a un et un seul $\overline{x}$ tel que $\text{symb}[\overline{x}] = x$;
on peut alors définir un ensemble attaché à une CF grammaire G et à toute phrase $\overline{x} \in \overline{L}(G)$;

$\overline{ASA}(\overline{x}, G) = \{\overline{\omega}_i \mid 0 < i \leq m, \overline{\omega}_0 = \overline{x}, \overline{\omega}_m = 0.1\ 0.2\ 0.3 \text{ et } \overline{\omega}_{j+1} \overset{R}{\Rightarrow} \overline{\omega}_j \text{ pour } 0 < j \leq m\}$;

on a : $\text{symb}[\overline{ASA}(\overline{x}, G)] = ASA(x, G) - \{\omega_0\}$;

l'ensemble $\overline{ASA}(\overline{x}, G)$ est l'ensemble ordonné de description ascendante pour la structure de $\overline{x}$,

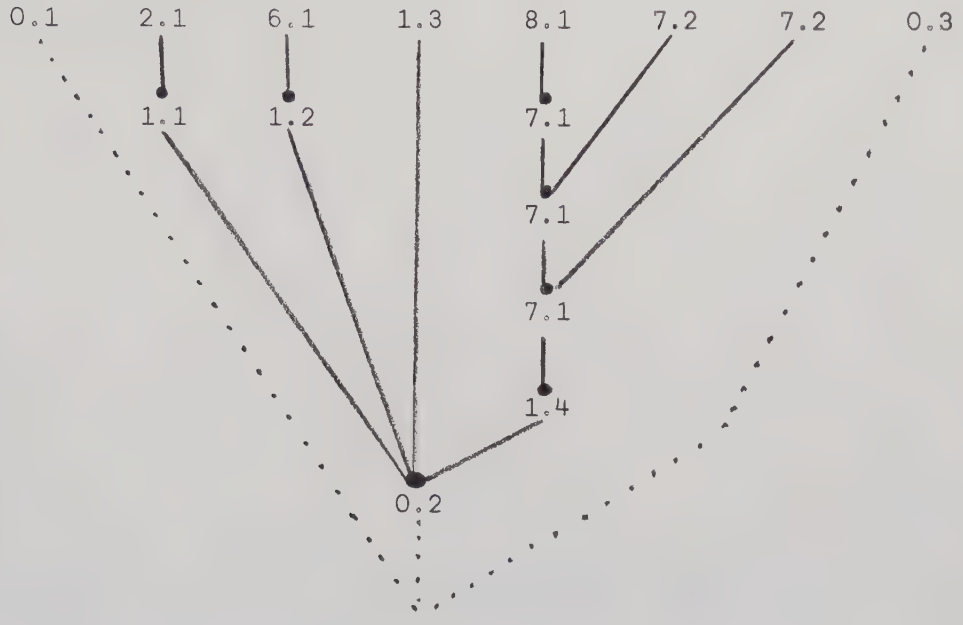
Exemple 5

Soit la grammaire G_o définie dans les exemples 1 et 3 soit $\overline{x}$ = 0.1 2.1 6.1 1.3 8.1 7.2 7.2 0.3 une phrase de $\overline{L}(G_o)$;
on a $\text{symb}[\overline{x}] = \text{symb}[0.1]\ \text{symb}[2.1]\ \text{symb}[6.1]$ ...
$\text{symb}[7.2]\ \text{symb}[0.3] = < + . ddd > = x \in L(G_o)$;
en dépit du caractère non injectif de symb pour la grammaire G_o, il n'existe pas $\overline{x}' \in \overline{L}(G_o)$ tel que :
$\text{symb}[\overline{x}] = \text{symb}[\overline{x}']$ et $0.1\ 0.2\ 0.3 \overset{*}{\Rightarrow} \overline{x}$ et $0.1\ 0.2\ 0.3 \overset{*}{\Rightarrow} \overline{x}'$ parce que G_o est non ambiguë
on a $\overline{ASA}(\overline{x}, G_o) =$

$\{\overline{\omega}_1 = 0.1\ 2.1\ 6.1\ 1.3\ 8.1\ 7.2\ 7.2\ 0.3$
$\overline{\omega}_2 = 0.1\ 1.1\ 6.1\ 1.3\ 8.1\ 7.2\ 7.2\ 0.3$
$\overline{\omega}_3 = 0.1\ 1.1\ 1.2\ 1.3\ 8.1\ 7.2\ 7.2\ 0.3$
$\overline{\omega}_4 = 0.1\ 1.1\ 1.2\ 1.3\ 7.1\ 7.2\ 7.2\ 0.3$
$\overline{\omega}_5 = 0.1\ 1.1\ 1.2\ 1.3\ 7.1\ 7.2\ 0.3$
$\overline{\omega}_6 = 0.1\ 1.1\ 1.2\ 1.3\ 1.4\ 0.3$
$\overline{\omega}_7 = 0.1\ 1.1\ 1.2\ 1.3\ 1.4\ 0.3$
$\overline{\omega}_8 = 0.1\ 0.2\ 0.3$ $\}$

il est aisé de vérifier d'après l'exemple 1 que pour $1 \leq i \leq 8$ on a $\text{symb}[\overline{\omega}_i] = \omega_{i-1} \in ASA(x, G)$

L'arbre syntaxique de $\overline{x}$ est l'arborescence construite à partir de l'ensemble $\overline{ASA}(\overline{x}, G_o)$; cette arboresence comporte 7 sommets non pendants



(en pointillé, le complément "naturel" de cet arbre).

mais en fait un analyseur syntaxique travaille à partir de $x \in L(G)$ et non pas de $\overline{x}$ $\overline{L}(G)$; on est donc conduit à définir l'ensemble

$\widehat{ASA}(x, G) = \{\hat{\omega}_i \mid 0 \leq i \leq m \;\; \hat{\omega}_o = x, \; \hat{\omega}_i = f[\overline{\omega}_i] \text{ pour } \overline{\omega}_i \in \overline{ASA}(\overline{x}, G)\}$;

f est définie de la façon suivante :

considérons $\overline{\omega}_{i+1}$ et $\overline{\omega}_i$ pour $0 < i < m$;

on a $\overline{\omega}_{i+1} \overset{R}{\rightarrow} \overline{\omega}_i$ c'est-à-dire que :

$\overline{\omega}_i = \overline{\overline{\omega}\,\overline{y}}$ et $\omega_{i+1} = \overline{c + e.d\,\overline{y}}$ avec $c + e.d \rightarrow \overline{\omega}$ et $symb|\overline{y}| \quad T^*$;

alors $f|\overline{\omega}_i| = \overline{\overline{\omega}\, symb|\overline{y}|}$ pour $0 < i < m$;

et $f|\overline{\omega}_m| = \overline{\omega}_m = 0.1\ 0.2\ 0.3$;

la justification intuitive d'un tel choix pour f est que, dans l'automate à pile qui constitue l'analyseur syntaxique, le contenu de la pile peut être écrit avec un alphabet à notre choix - par exemple C(G) au lieu de V usuellement - alors que le ruban d'entrée contient un facteur droit d'une phrase terminale ce qui impose T comme vocabulaire.

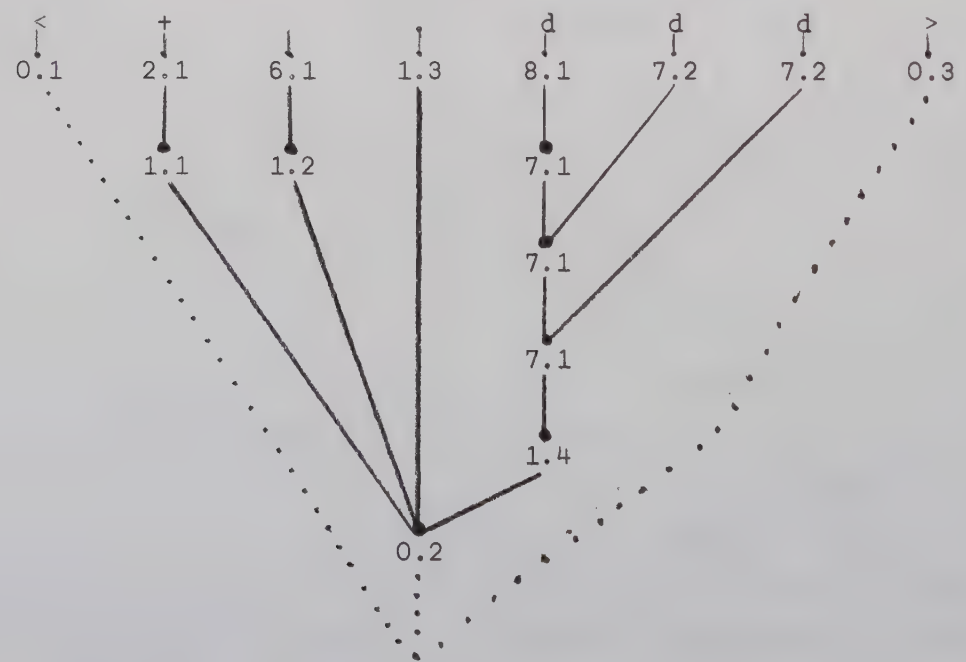
Exemple 6

Soit la grammaire G_o définie dans les exemples 1 et 3;
soit x = < + . ddd > une phrase de $L(G_o)$;
on a $\widehat{ASA}(x, G_o)$ =

$\{\hat{\omega}_o$ = < + . ddd >
$\hat{\omega}_1$ = 0.1 2.1 . ddd >
$\hat{\omega}_2$ = 0.1 1.1 . ddd >
$\hat{\omega}_3$ = 0.1 1.1 6.1 . ddd >
$\hat{\omega}_4$ = 0.1 1.1 1.2 1.3 8.1 dd >
$\hat{\omega}_5$ = 0.1 1.1 1.2 1.3 7.1 7.2 d >
$\hat{\omega}_6$ = 0.1 1.1 1.2 1.3 7.1 7.2 >
$\hat{\omega}_7$ = 0.1 1.1 1.2 1.3 1.4 >
$\hat{\omega}_8$ = 0.1 0.2 0.3 $\}$

il est aisé de vérifier d'après l'exemple 5 que pour
$1 \leq i \leq 8$, on a $\hat{\omega}_i = f[\overline{\omega}_i]$ pour $\overline{\omega}_i \in \overline{ASA}(\overline{x}, G_o)$

l'arbre syntaxique de x est l'arborescence construite à partir de l'ensemble $\widehat{ASA}(x, G_o)$; cette arborescence comporte $7 + |\overline{x}|$ sommets non pendants



(en pointillé, le complément "naturel" de cet arbre)

On constate aisément que cet arbre est entièrement équivalent à celui de l'exemple 1.

On peut maintenant définir un analyseur syntaxique ascendant pour un langage L(G) engendré par une CF grammaire G non ambiguë :
c'est un algorithme qui a pour donnée une chaîne de longueur finie $x \in T^*$, qui lit x strictement de gauche à droite en tentant de calculer simultanément (*) les éléments de $\widehat{ASA}(x, G)$ et qui s'arrête après un nombre fini d'opérations pour une des deux raisons suivantes :

1) $x \in L(G)$ auquel cas le résultat est $\widehat{ASA}(x, G)$;

2) $x \notin L(G)$

en indiquant si l'arrêt est dû à la cause 1 ou à la cause 2.

La définition est équivalente à celle du chapitre 1 car il est aisé de construire ASA(x, G) à partir de $\widehat{ASA}(x, G)$;
pour terminer, explicitons ce qu'est cet analyseur syntaxique sous forme d'un automate à pile AP(G) = (Q, I, E, P, S) où
Q est l'ensemble des états : initial 9_I, courant 9, final 9_F;
E est le ruban d'entrée d'alphabet T;
P est la pile, d'alphabet $\{\#\} \cup C(G) \cup E(G)$; # sert à marquer le fond de P
S est le ruban de sortie, d'alphabet $\{\#\} \cup C(G) \cup T$; # sépare les phrases sur S
I est un ensemble d'instructions de type :
$(9, p, e) \rightarrow (9', p', s')$
avec 9 et $9' \in Q$; p et p' sont des chaînes de longueur finie sur l'alphabet de P, e est une chaîne de longueur finie sur l'alphabet de E, s' est une chaîne de longueur finie sur l'alphabet de S; une chaîne de longueur nulle est notée ε; après exécution d'une telle instruction les rubans E et S progressent de |e| et |s'| respectivement;
l'ensemble I des instructions de AP(G) est alors

$(9_I, \varepsilon, \vdash) \rightarrow (9, \# \; 0.1, \varepsilon)$ [0]

$(9, c + e.d, a) \rightarrow (9, c + e.d\ f + h.g, \varepsilon)$ [1]

$(9, c + e.d, \varepsilon) \rightarrow (9, c + e.d\ f + h/g, \varepsilon)$ [2]

$(9, i + k.1\ i\ k.2 \ldots i + k.n(i + k), \varepsilon) \rightarrow$
$(9, i.0, \overline{\varphi}\ i + k.1\ i + k.2 \ldots i + k.n(i + k)\ y\ \#)$ [3]

$(9, i.0, \varepsilon) \rightarrow (9, f + h.g, \varepsilon)$ [4]

$(9, \# \; 0.1\ 0.2\ 0.3, \varepsilon) \rightarrow (9_F, \ , 0.1\ 0.2\ 0.3\ \#)$ [5]

(*) plus exactement en tenant simultanément de calculer

avec les conditions suivantes :

instruction [1] : $d < n(c + e)$, $\forall f + h.g \in a$;
instruction [2] : $d < n(c + e)$, $\forall f + h.g \in \varepsilon$;
instruction [3] : $\overline{\varphi}$ est la chaîne contenue dans P sous la chaîne $i + k.1\ i + k.2 \ldots i + k.n(i + k)$ et y est la chaîne contenue dans la partie non encore lue de S;
instruction [4] : pour $\forall f + h.g \in i.0$

quand on soumet $x \in L(G)$ à AP(G), celui-ci termine son calcul dans l'état 9_F avec les éléments de $\hat{ASA}(x, G)$ écrits sur S séparés par des occurrences du symbole #;
les instructions [1], [2], [4] rendent AP(G) non déterministe sauf dans le cas où symb est un isomorphisme; de la même façon, l'algorithme de base d'analyse ascendante avec un vocabulaire de pile V n'est pas déterministe et l'emploi de considérations de contexte est la démarche qui a été utilisée pour obtenir des algorithmes d'analyse ascendante déterministes tels que ceux à contexte borné [LOE 70] nous allons expliquer une démarche similaire pour rendre déterministe l'algorithme d'analyse ascendante avec un vocabulaire de pile construit à partir de C(G). Rappelons que l'emploi d'un alphabet de pile composé d'éléments de C(G) (et aussi de E(G) mais ceci importe peu car N - {Z} pourrait lui être substitué n'étaient les questions de notations déjà évoquées plus haut) laisse espérer des résultats meilleurs que l'alphabet classique V pour l'obtention du déterminisme parce qu'un symbole de C(G) contient plus d'information que son image homomorphe dans V. Le second fournit un nom de symbole alors que le premier fournit un nom et une position d'occurrence dans les productions et permet donc la construction de contextes à gauche plus riches à longueur égale.

CHAPITRE 3

GENERATION DES CHAINES DE CONTEXTE

GENERALITES

Intuitivement parlant, il s'agit de calculer les divers contextes de longueurs bornées à gauche et à droite dans lesquels se produisent les différentes occurrences de tout symbole $\alpha \in V$; on peut ainsi espérer pouvoir différencier les diverses occurrences de tout symbole α en consultant les contextes associés et donc rendre déterministe le fonctionnement d'un analyseur ascendant.

Soit deux entiers $m \geq 1$ et $q \geq 0$, on se propose, pour $\forall i + k.j \in C(G)$ de calculer les chaînes

$$\overline{\sigma}_t[i + k.j] = \overline{\varphi}_t[i + k.j]\ i + k.j\ \overline{z}_t[i + k.j] \text{ avec } 1 \leq t \leq \overline{ns}[i + k.j]$$

sous les conditions suivantes :

soit tous les $\overline{\varphi}$ et $\overline{z}$ tels que :

$$0.1\ 0.2\ 0.3 \overset{R^*}{\Rightarrow} \overline{\varphi}\ i + k.j\ \overline{z} \text{ avec } \mathrm{symb}[\overline{\varphi}\ i + k.j] \in V^* \text{ et } \mathrm{symb}[\overline{z}] \in T^*$$

alors $\overline{\varphi}_t[i + k.j] = \overline{\varphi} : m$

et $\overline{z}_t[i + k.j] = q' : \overline{z}$, q' étant tel que

$$| \ \mathrm{symb}[i + k.j\ \overline{z}_t\ [i + k.j]] \ | = q + 1$$

il y a un nombre fini $\overline{ns}[i + k.j]$ de telles chaînes $\overline{\sigma}_t[i + k.j]$ distinctes au moins pour la raison que le vocabulaire C(G) est fini ainsi que la longueur de ces chaînes;

pour obtenir les contextes nécessaires au fonctionnement d'un analyseur ascendant, il faut modifier les chaînes $\overline{\sigma}_t[i + k.j]$ pour obtenir des chaînes

$$\hat{s}_t[i + k.j] = \overline{\varphi}_t[i + k.j]\ cl[i + k.j]\ r_t[i + k.j]$$

avec $1 \leq t \leq \hat{ns}[i + k.j] \leq ns[i + k.j]$

$cl[i + k.j]$ = si $\mathrm{symb}[i + k.j] \in N - \{Z\}$ alors f.0 tel que $i + k.j \in f.0$ sinon 1 : $\mathrm{symb}[i + k.j\ \overline{z}_t]i + k.j]]$,

$r_t[i + k.j] = \mathrm{symb}[i + k.j\ \overline{z}_t[i + k]] : | \mathrm{symb}[i + k.j\ \overline{z}_t[i + k.j]] | - 1$;

il y a $\hat{ns}[i + k.j]$ chaînes $\hat{s}_t[i + k.j]$ ainsi construites;

on a $\hat{ns}[i + k.j] \leq \overline{ns}[i + j.j]$ parce qu'on applique l'homomorphisme symb à la partie droite des chaînes $\overline{\sigma}_t[i + k.j]$ et que donc, plusieurs chaînes $\overline{\sigma}_t[i + k.j]$ peuvent avoir même image $\hat{s}_{t'}[i + k.j]$;

Exemple 7

Soit la grammaire G_0 définie dans les exemples 1 et 3, les ensembles de chaînes $\overline{\sigma}_t[i + k.j]$ et $\hat{s}_t[i + k.j]$ sont donnés dans le tableau suivant

	← $\overline{\sigma}_t[i + k.j]$ →			← $\hat{s}_t[i + k.j]$ →		
t	$\overline{\varphi}_t[i + k.j]$	$i + k.j$	$\overline{z}_t[i + k.j]$	$\overline{\varphi}_t[i + k.j]$	$c\ell[i + k.j]$	$r_t[i + k\ j]$
1	$\underline{\varepsilon}$	0.1	2.1	$\underline{\varepsilon}$	<	+
2	$\underline{\varepsilon}$	0.1	3.1	$\underline{\varepsilon}$	<	-
3	$\underline{\varepsilon}$	0.1	4.1 6.1 1.3	$\underline{\varepsilon}$	<	.
4	$\underline{\varepsilon}$	0.1	4.1 8.1	$\underline{\varepsilon}$	<	d
1	0.2	0.3	$\underline{\varepsilon}$	0.2	>	ε
1	1.2	1.3	6.1 0.3	1.2	.	>
2	1.2	1.3	8.1	1.2	.	d
1	0.1	2.1	6.1 1.3	0.1	+	.
2	0.1	2.1	8.1	0.1	+	d
1	0.1	3.1	6.1 1.3	0.1	-	.
2	0.1	3.1	8.1	0.1	-	d
1	0.1	4.1	6.1 1.3 6.1 0.3	0.1	.	>
2	0.1	4.1	8.1 1.3	0.1	d	.
3	0.1	4.1	8.1 7.2	0.1	d	d
4	0.1	4.1	6.1 1.3 8.1	0.1	.	d
1	1.1	6.1	1.3 6.1 0.3	1.1	.	>
2	1.3	6.1	0.3	1.3	>	ε
3	1.1	6.1	1.3 8.1	1.1	.	d
1	7.1	7.2	1.3	7.1	d	.
2	7.1	7.2	7.2	7.1	d	d
3	7.1	7.2	0.3	7.1	d	>
1	1.1	8.1	1.3	1.1	d	
2	1.1	8.1	7.2	1.1	d	d
3	1.3	8.1	0.3	1.3	d	>
4	1.3	8.1	7.2	1.3	d	d
1	1.1	5.1	1.3	1.1	7.0	.
2	1.3	5.1	0.3	1.3	7.0	>
1	1.1	7.1	7.2	1.1	7.0	d
2	1.3	7.1	7.2	1.3	7.0	d
1	1.1	1.2	1.3	1.1	5.0	.
1	1.3	1.4	0.3	1.3	5.0	>
1	0.1	1.1	6.1 1.3	0.1	2.0	.
2	0.1	1.1	8.1	0.1	2.0	d
1	0.1	0.2	0.3	0.1	1.0	>

On voit que pour $\forall\ i + k.j \in C(G_o)$, $\hat{ns}[i + k.j] = \overline{ns}[i + k.j]$

pour chaque $i + k.j \in C(G)$ on définit l'ensemble des chaînes de contexte :

$$S[i + k.j] = \{\hat{s}_t[i + k.j],\ 1 \leq t \leq \hat{ns}[i + k.j]\};$$

sans répétition dans l'énumération c'est-à-dire que :

$$t_1 \neq t_2 \Leftrightarrow \hat{s}_{t_1}[i + k.j] \neq \hat{s}_{t_2}[i + k.j];$$

il s'agit maintenant de savoir comment on constuit les chaînes $\bar{\sigma}_t[i + k.j]$

ALGORITHME POUR GENERER LES CHAINES DE CONTEXTE

Cet algorithme dérive directement de l'algorithme indiqué par J. LOECKX pour construire les chaînes de contexte nécessaires au fonctionnement d'un analyseur à contexte borné [LOE 70];

Soit $\mathscr{A}$ une relation binaire sur C(G) se définissant comme suit :

$$i + k.j \,\mathscr{A}\, f + h.g \Leftrightarrow \exists \bar{\varphi} \text{ et } \bar{\Psi} \text{ tels que } 0.1\ 0.2\ 0.3 \overset{*}{\rightarrow} \bar{\varphi}\ f + h.g\ i + k.j\ \bar{\Psi}$$

et définissant un sous-ensemble de C(G) x C(G);

pour calculer cette relation, il est nécessaire d'en définir deux autres :

$\mathscr{S}$ est une relation binaire sur C(G) se définissant comme suit :

$$i + k.1 \,\mathscr{S}\, f + h.g \Leftrightarrow F + h.g \in i.0$$

il est immédiat de construire $\mathscr{S}^+$ par fermeture transitive de $\mathscr{S}$; $\mathscr{S}^+$ définit un sous-ensemble de $\{i + k.j \mid i + k.j \in C(G) \text{ et } j = 1\} \times C(G)$;

$\mathscr{D}$ est une relation binaire sur C(G), définie comme suit :

$$i + k.n(i + k)\, \mathscr{D}\, c + e.d \Leftrightarrow c + e.d \in i.0;$$

il est immédiat de construire $\mathscr{D}^+$ par fermeture transitive de $\mathscr{D}$; $\mathscr{D}^+$ définit un sous-ensemble de $\{i + k.j \mid i + k.j \in C(G) \text{ et } j = n(i + k)\} \times C(G)$;

le calcul de la relation $\mathscr{R}$ est alors :

$i + k.j + 1\, \mathscr{R}\, i + k.j$ pour $\forall i + k.j \in C(G)$ tel que $j < n(i + k)$

$c + e.1\, \mathscr{R}\, i + k.j$ pour $\forall i + k.j \in C(G)$ tel que $j < n(i + k)$ et $c + e.1 \in C(G)$ tel que $c + e.1\, \mathscr{S}^+\, i + k.j + 1$

$i + k.j + 1\, \mathscr{R}\, f + h.n(f + h)$ pour $\forall i + k.j + 1 \in C(G)$ tel que $j < n(i + k)$ et $\forall f + h.n(f + h)$ tel que $f + h.n(f + h)\, \mathscr{D}^+\, i + k.j$

$c + e.1\, \mathscr{R}\, f + h.n(f + h)$ pour $\forall c + e.1$ et $f + h.n(f + h) \in C(G)$ tels que $c + e.1\, \mathscr{S}^+\, i + k.j + 1$ et $f + h.n(f + h)\, \mathscr{D}^+\, i + k.j$ pour $\forall i + k.j \in C(G)$ avec $j < n(i + k)$;

On peut trouver un algorithme pratique pour calculer la fermeture transitive d'une relation binaire à l'aide d'une matrice d'éléments booléens dans [GRI 69];

Exemple 8

Soit la grammaire G_o définie dans les exemples 1 et 3; la relation $\mathcal{P}^+$ est indiquée sous forme d'une matrice booléenne MP

	0.2	1.1	1.2	1.4	5.1	7.1
1.1	1					
2.1	1	1				
3.1	1	1				
4.1	1	1				
5.1			1	1		
6.1			1	1		
7.1			1	1	1	1
8.1			1	1	1	1

On a MP(i + k.1, c + e.d) = 1 $\Leftrightarrow$ i + k.1 $\mathcal{P}^+$ c + e.d;
$\mathcal{D}^+$ est indiqué sous la forme d'une matrice booléenne MD

	0.2	1.1	1.2	1.4	5.1	7.1
1.4	1					
2.1		1				
3.1		1				
4.1		1				
5.1	1		1	1		
6.1	1		1	1		
7.2	1		1	1	1	1
8.1	1		1	1	1	1

On a MD(i + k.n(i + k), c + e.d) = 1 $\Leftrightarrow$ i + k.n(i + k) $\mathcal{D}^+$ c + e.d;

$\mathfrak{R}$ est indiquée sous forme d'une matrice booléenne MA

	0.1	0.2	0.3	1.1	1.2	1.3	1.4	2.1	3.1	4.1	5.1	6.1	7.1	7.2	8.1
0.1															
0.2	1														
0.3		1					1				1	1		1	1
1.1	1														
1.2				1				1	1	1					
1.3					1						1	1		1	1
1.4						1									
2.1	1														
3.1	1														
4.1	1														
5.1				1		1		1	1	1					
6.1				1		1		1	1	1					
7.1				1		1		1	1	1					
7.2													1	1	1
8.1				1		1		1	1	1					

On a $MA[i + k.j, c + e.d] = 1 \Leftrightarrow i + k.j \mathfrak{R} c + e.d$

il est maintenant possible de décrire un algorithme permettant, pour chaque $i + k.j \in C(G)$, de générer l'ensemble de chaînes $S[i + k.j]$; cet algorithme est exécuté par un automate à pile très proche de celui décrit dans [LOE 70]

$GE(i + k.j, G) = (Q, I, P, S_L, S_R)$ où

Q est l'ensemble des états : initial q_I, courants q_L et q_R, final q_F;
P est une pile modifiée dont le fond et le sommet dans l'état q_L sont le sommet et le fond dans l'état q_R; l'alphabet de P est

$\{\#\} \cup C(G) \cup E(G)$;

S_L est un ruban de sortie d'alphabet C(G)
S_R est un ruban de sortie d'alphabet C(G)
I est un ensemble d'instructions de type :

$$(q, p) \rightarrow (q', p', s_L, s_R)$$

avec q et $q' \in Q$; p et p' sont des chaînes de longueur finie sur l'alphabet de P, s_L et s_R sont des chaînes de longueur finie sur les alphabets de S_L et S_R; une chaîne de longueur nulle est notée ε; après exécution d'une telle instruction les rubans S_L et S_R progressent de $|s_L|$ et $|s_R|$ respectivement; au point de vue notation, le haut de P sera considéré être à gauche dans l'état q_L et à droite dans l'état q_R;
l'ensemble I des instructions de GE(i + k.j, G) est alors :

$(q_I, \varepsilon) \rightarrow (q_L, \# \; i + k.j \; \#, \varepsilon, \varepsilon)$ [0]
$(q_L, \# \; c + e.1 \; c + e.2 \ldots c + e.n(c + e)) \rightarrow (q_L, \# \; c.0, \varepsilon, \varepsilon)$ [1]
$(q_R, c + e.1 \; c + e.2 \ldots c + e.n(c + e) \; \#) \rightarrow (q_R, c.0 \; \#, \varepsilon, \varepsilon)$ [2]
$(q_L, \# \; c.0 \; \#) \rightarrow (q_L, \# \; f + h.g \; \#, \varepsilon, \varepsilon)$ [3]
$(q_R, \# \; c.0 \; \#) \rightarrow (q_R, \# \; f + h.g \; \#, \varepsilon, \varepsilon)$ [4]
$(q_L, \# \; c.0 \; f + h.g) \rightarrow (q_L, \# \; r + t.s \; f + h.g, \varepsilon, \varepsilon)$ [5]
$(q_R, f + h.g \; c.0 \; \#) \rightarrow (q_R, f + h.g \; r + t.s \; \#, \varepsilon, \varepsilon)$ [6]
$(q_L, \# \; f + h.g) \rightarrow (q_L, \# \; c + e.d \; f + h.g, c + e.d, \varepsilon)$ [7]
$(q_R, f + h.g \; \#) \rightarrow (q_R, f + h.g \; c + e.d \; \#, \varepsilon, c + e.d)$ [8]
$(q_L, \varepsilon) \rightarrow (q_R, \varepsilon, \varepsilon, \varepsilon)$ [9]
$(q_R, \varepsilon) \rightarrow (q_L, \varepsilon, \varepsilon, \varepsilon)$ [10]
$(q_L, \# \; 0.1 \; 0.2 \; 0.3 \; \#) \rightarrow (q_F, \varepsilon, \varepsilon, \varepsilon)$ [11]
$(q_R, \# \; 0.1 \; 0.2 \; 0.3 \; \#) \rightarrow (q_F, \varepsilon, \varepsilon, \varepsilon)$ [12]
$(q_L, \varepsilon) \rightarrow (q_F, \varepsilon, \varepsilon, \varepsilon)$ [13]
$(q_R, \varepsilon) \rightarrow (q_F, \varepsilon, \varepsilon, \varepsilon)$ [14]

avec les conditions suivantes :

instructions [3] et [4] : $\forall f + h.g \in c.0$
instruction [5] : $\forall r + t.s \in c.0$ tel que $f + h.g \not\propto r + t.s$;
instruction [6] : $\forall r + t.s \in c.0$ tel que $r + t.s \not\propto f + h.g$;
instruction [7] : $\forall f + h.g$ et $c + e.d$ tels que $g < n(f + h)$,
$d < n(c + e)$ et $f + h.g \not\propto c + e.d$;
instruction [8] : $\forall f + h.g$ et $c + e.d$ tels que $g < n(f + h)$,
$c + e.d \not\propto f + h.g$ et $\text{symb}[c + e.d] \in T$;

instruction [9] : dès que S_L contient m symboles ou le symbole 0.1
instruction [10] : dès que S_R contient q' symboles ou le symbole 0.3
instructions [11] et [12] : aucune génération n'est plus possible, ni à gauche, ni à droite;
instructions [13] et [14] : dès que la chaîne écrite sur S_1 a la longueur m et celle écrite sur S_2 a la longueur q';
la génération effective intervient dans les instructions [7] et [8];

Exemple 9

Le tableau ci-dessous indique la suite des descriptions instantanées de GE(6.1, G_o) pour m = q = 1

numéro de description	état	P	S_L	S_R	numéro de l'instruction applicable	numéros des descriptions obtenues après application de l'instruction applicable
1	q_I				0	2
2	q_L	# 6.1 #			1	3
3	q_L	# 5.0 #			3	4, 17
4	q_L	# 1.2 #			7	5
5	q_L	# 1.1 1.2 #	1.1		9	6
6	q_R	# 1.1 1.2 #	1.1		8	7
7	q_R	# 1.1 1.2 1.3 #	1.1	1.3	8	8, 15
8	q_R	# 1.1 1.2 1.3 6.1 #	1.1	1.3 6.1	4	9
9	q_R	# 1.1 1.2 1.3 5.0 #	1.1	1.3 6.1	6	10
10	q_R	# 1.1 1.2 1.3 1.4 #	1.1	1.3 6.1	2	11
11	q_R	# 1.0 #	1.1	1.3 6.1	4	12
12	q_R	# 0.2 #	1.1	1.3 6.1	8	13
13	q_R	# 0.2 0.3 #	1.1	1.3 6.1 0.3	14	14
14	q_F	# 0.2 0.3 #	1.1	1.3 6.1 0.3	-	-
15	q_R	# 1.1 1.2 1.3 8.1 #	1.1	1.3 8.1	14	16
16	q_F	# 1.1 1.2 1.3 8.1 #	1.1	1.3 8.1	-	-
17	q_L	# 1.4 #			7	18
18	q_L	# 1.3 1.4 #	1.3		9	19
19	q_R	# 1.3 1.4 #	1.3		8	20
20	q_R	# 1.3 1.4 0.3 #	1.3	0.3	14	21
21	q_F	# 1.3 1.4 0.3 #	1.3	0.3	-	-

On voit qu'il y a trois copies du générateur qui ont produit les chaînes suivantes sur S_L et S_R

1.1 et 1.3 6.1 0.3
1.1 et 1.3 8.1
1.3 et 0.3

d'où les chaînes

$\overline{\sigma}_1[6.1] = 1.1\ 6.1\ 1.3\ 6.1\ 0.3$
$\overline{\sigma}_2[6.1] = 1.1\ 6.1\ 1.3\ 8.1$
$\overline{\sigma}_3[6.1] = 1.3\ 6.1\ 0.3$

intuitivement, elle procède comme suit :
supposons qu'avant d'appliquer l'une de ces deux instructions, P contienne $\overline{\varphi} \in \overline{PHA}(G)$; on veut concaténer à $\overline{\varphi}$ soit un symbole $\overline{\alpha}$ à gauche tel que :

1) $\overline{\alpha}\,\overline{\varphi} \in \overline{PHA}(G)$, $\text{symb}[\overline{\alpha}\ k : \overline{\varphi}] \in V^*$, $\text{symb}[\overline{\varphi} : |\overline{\varphi}| - k] \in T^*$;
soit un symbole $\overline{\beta}$ à droite tel que :

2) $\overline{\varphi}\,\overline{\beta} \in \overline{PHA}(G)$, $\text{symb}[k : \overline{\varphi}] \in V^*$, $\text{symb}[\overline{\varphi} : |\overline{\varphi}| - k\ \overline{\beta}] \in T^*$;
ce qui exprime que $\overline{\alpha}$ doit être un nouveau symbole du contexte gauche et symb $[\overline{\beta}]$ un nouveau symbole du contexte droit;
pour le contexte gauche :
$\overline{\alpha}\,\overline{\varphi} \in \overline{PHA}(G) \Rightarrow \overline{\alpha}\,\overline{\varphi} \in \overline{PH}(G) \Rightarrow (1 : \overline{\varphi})\,\mathcal{R}\,\overline{\alpha}$ par définition; de plus la condition 1 entraîne que $\overline{\alpha}$ ne peut être que de la forme c + e.d avec $d < n(c + e)$;
pour le contexte droit :
$\overline{\varphi}\,\overline{\beta} \in \overline{PHA}(G) \Rightarrow \overline{\varphi}\,\overline{\beta} \in \overline{PH}(G) \Rightarrow \overline{\beta}\,\mathcal{R}(\overline{\varphi} : 1)$ par définition; de plus la condition 2 entraîne que symb $[\overline{\beta}] \in T$;
ce qui précède doit aider à comprendre le fonctionnement de l'automate générateur; que celui-ci génère effectivement les chaînes $\overline{\varphi}_t[i + k.j]$ sur le ruban S_L et les chaînes $\overline{z}_t[i + k.j]$ sur le ruban S_R est démontré dans la thèse de doctorat de J. LOECKX [LOE 68];
ce générateur est fortement indéterministe et donc sa mise en oeuvre nécessite, chaque fois qu'une situation indéterministe se présente au cours d'un calcul, d'en faire autant de copies que nécessaire; à la fin du calcul des chaînes de contexte pour $i + k.j \in C(G)$ il y a $\overline{ns}[i + k.j]$ copies du générateur initial; ayant ainsi obtenu de GE(i + k.j, G) toutes les chaînes $\overline{\varphi}_t[i + k.j]$ et $\overline{z}_t[i + k.j]$ il est immédiat de construire les chaînes $\hat{s}_t[i + k.j]$ et l'ensemble $S[i + k.j]$;
nous définissons encore, pour chaque $f.0 \in D(G)$ l'ensemble de chaînes :

$\bar{u}_h[f.0] = f + h.1\ f + h.2 \ldots f + h.n(f + h)$ pour $0 \leq h < p(A_{f.0})$;

soit $U[f.0] = \{\bar{u}_h[f.0] \mid 0 \leq h < p(A_{f.0})\}$;

soit enfin

$$CX(G, m, k) = \{\bigcup_{i + k.j \in C(G)} S[i + k.j]\} \bigcup \{\bigcup_{f.0 \in D(G)} U[f.0]\}$$

ce dernier ensemble de chaînes va servir de base à la construction d'une table de décision pour un analyseur ascendant déterministe.

Exemple 10

L'ensemble $CX(G_0, 1, 1)$ se compose de tous les ensembles $S[i + k.j]$ composés de toutes les chaînes $\hat{s}_t[i + k.j]$ et de tous les ensembles $U(f.0)$ de toutes les chaînes $u_h(f.0)$
entre autres :

$S[0 + 0.1] = \{\hat{s}_1[0 + 0.1] = < +,\ \hat{s}_2[0 + 0.1] = < -,$
$\hat{s}_3[0 + 0.1] = < .,\ \hat{s}_4[0 + 0.1] = < d\}$

$S[7 + 1.1] = \{\hat{s}_1\ 7[+ 1.1] = 1.1\ d\ .,\ \hat{s}_2[7 + 1.1] = 1.1\ dd,$
$\hat{s}_3[7 + 1.1] = 1.3\ d >,\ \hat{s}_4[7 + 1.1] = 1.3\ dd\}$

$S[1 + 0.4] = \{\hat{s}_1[1 + 0.4] = 1.3\ 5.0 >\}$

$S[0 + 0.2] = \{\hat{s}_1[0 + 0.2] = 0.1\ 1.0 >\}$

$U[0.0] = \{\bar{u}_0[0.0] = 0 + 0.1\ 0 + 0.2\ 0 + 0.3\}$

$U[2.0] = \{\bar{u}_0[2.0] = 2 + 0.1,\ \bar{u}_1[2.0] = 2 + 1.1,\ \bar{u}_2[2.0] = 2 + 2.1\}$

$U[7.0] = \bar{u}_0[7.0] = 7 + 0.1\ 7 + 0.2,\ \bar{u}_1[7.0] = 7 + 1.1\}$

CHAPITRE 4

RECHERCHE DU DETERMINISME DE L'ANALYSEUR

L'ensemble CX(G, m, q) écrit sur l'alphabet $C(G) \cup E(G)$ permet-il de construire directement un analyseur déterministe ?

La réponse est positive si cet ensemble de chaînes vérifie la condition suivante :

$$\mathcal{C}_1 \begin{cases} \text{pour } \forall i + k.j \text{ et } i' + k'.j' \in C(G) \\ i + k.j \neq i' + k'.j' \Leftrightarrow S[i + k.j] \cap S[i' + k'.j'] = \emptyset \end{cases}$$

car on est alors capable de distinguer, pour tout symbole de W, toutes ses différentes occurrences par comparaison de son contexte avec les chaînes de CX(G, m, q);
cependant cette condition $\mathcal{C}_1$ est beaucoup trop restrictive; nous allons en déduire un algorithme pour les cas où elle n'est pas satisfaite.

Exemple 11

CX(G_0, 1, 1) écrit sur l'alphabet $C(G_0)$ satisfait la condition $\mathcal{C}_1$; par contre la grammaire suivante, citée en exemple dans [BOR 71] p.

$$G_9 = (N_9, T_9, R_9, Z)$$

avec $N_9 = \{Z, S, T\}$; $T_9 = \{<, >, a, b, c\}$;

$R_9 = \{Z_{0.0} \rightarrow {<}_{0.1}\ S_{0.2}\ {>}_{0.3}$

$S_{1.0} \rightarrow a_{1.1}\ S_{1.2} \mid a_{2.1}\ T_{2.2}\ b_{2.3}\ S_{2.4} \mid c_{3.1}$

$T_{4.0} \rightarrow a_{4.1}\ T_{4.2}\ b_{4.3}\ T_{4.4} \mid c_{5.1}\}$

ne satisfait pas à la condition $\mathcal{C}_1$ car visiblement

$$S[1.1] \cap S[2.1] \cap S[4.1] \ni 0.1\ a^n \quad \text{pour tout } n$$

et par conséquent quelque soient m et q;
de même la grammaire suivante, dérivant de contre-exemples construits par M. BRANQUART (M.B.L.E. Bruxelles)

$$G_7 = (N_7, T_7, R_7, Z)$$

avec $N_7 = \{Z, S, A\}$; $T_7 = \{<, >, x, y, a\}$

$R_7 = \{Z_{0.0} \rightarrow {<}_{0.1}\ S_{0.2}\ {>}_{0.3}$

$S_{1.0} \rightarrow x_{1.1}\ A_{1.2}\ a_{1.3} \mid y_{2.1}\ A_{2.2}$

$A_{3.0} \rightarrow a_{3.1}\ A_{3.2} \mid a_{4.1}\}$

ne satisfait pas à la condition $\mathcal{C}_1$ car visiblement

$$S[3.1] \cap S[4.1] \supset \{3.1^n aa > \text{ pour tout } n$$

et par conséquent, quelque soient m et q.

Pour noter que l'ensemble de chaînes CX(G, m, q) est écrit sur l'alphabet $C(G) \cup E(G)$, on écrira cet ensemble comme suit :

$$\frac{CX(G, m, q)}{C(G) \cup E(G)};$$

de la même façon, pour noter qu'une chaîne, par exemple $\hat{s}_t[i + k.j]$, de cet ensemble est écrite sur ce même alphabet, on écrira

$$\frac{\hat{s}_t[i + k.j]}{C(G) \cup E(G).};$$

on notera $C^{(0)}(G) = C(G)$ et $CX^{(0)}(G, m, q) = CX(G, m, q)$;
soit une relation binaire $\mathcal{R}_1$ définie sur un ensemble $C^{(a)}(G)$ en utilisant un ensemble $CX^{(a)}(G, m, q)$:

$$\begin{cases} \forall i + k.j \text{ et } i' + k'.j' \in C^{(a)}(G) \\ i + k.j \,\mathcal{R}_1\, i' + k'.j' \Leftrightarrow \\ \dfrac{S[i + k.j]}{C^{(a)}(G) \cup E(G)} \cap \dfrac{S[i' + k'.j']}{C^{(a)}(G) \cup E(G)} \neq \emptyset, \; j < n(i + k) \\ \text{et } j' < n(i' + k') \end{cases}$$

Exemple 12

a) aucun couple de $C(G_0)$ n'est dans la relation $\mathcal{R}_1$ puisque $\mathcal{C}_1$ est satisfaite par $CX(C_0, 1, 1)$;

b) $\mathcal{R}_1$ n'est pas une condition d'équivalence : en effet soit

$S[1.1] = 0.1\ ab,\ 0.1\ ac \qquad 1 < n(1)$
$S[2.1] = 0.1\ ab,\ 0.1\ ad \qquad 1 < n(2)$
$S[3.1] = 0.1\ ae,\ 0.1\ ad \qquad 1 < n(3)$

pour une grammaire G quelconque;

On a 1.1 $\mathfrak{R}_1$ 2.1 et 2.1 $\mathfrak{R}_1$ 3.1 mais non pas 1.1 $\mathfrak{R}_1$ 3.1

c) pour G_9 on a 1.1 $\mathfrak{R}_1^+$ 2.1

1.1 $\mathfrak{R}_1^+$ 4.1

2.1 $\mathfrak{R}_1^+$ 4.1

donc un élément de $C^{(1)}(G_q) = C^{(0)}(G_q)/\mathfrak{R}_1^+$ est {1.1, 2.1, 4.1}

d) pour G_7 on n'a pas 3.1 $\mathfrak{R}_1$ 4.1

les conditions $j < n(i + k)$ et $j' < n(i' + k')$ signifient qu'on met dans la relation $\mathfrak{R}_1$ des symboles ne faisant pas varier de façons différentes la longueur de la pile d'un analyseur; la relation $\mathfrak{R}_1$ est visiblement réflexive et symétrique; $\mathfrak{R}_1^+$ obtenue par fermeture transitive de $\mathfrak{R}_1$ est donc une relation d'équivalence sur $C^{(a)}(G)$; par définition

$$C^{(a+1)}(G) = C^{(a)}(G)/\mathfrak{R}_1^+. \quad \text{et} \quad CX^{(a)}(G, m, q) = \frac{CX(G, m, q)}{C^{(a)}(G) \cup E(G)}$$

on définit aussi une condition de déterminisme qui est une généralisation de $\mathfrak{C}_1$, à savoir

$$\mathfrak{C} \begin{cases} \forall i + k.j \text{ et } i' + k'.j' \in C^{(a)}(G) \\ i + k.j \neq i' + k'.j' \Leftrightarrow \\ \dfrac{S[i + k.j]}{C^{(a)}(G) \cup E(G)} \cap \dfrac{S[i' + k'.j']}{C^{(a)}(G) \cup E(G)} = \emptyset \end{cases}$$

Un algorithme permettant de construire $C^{(a)}(G)$ et $CX^{(a)}(G, m, q)$ satisfaisant $\mathfrak{C}$ quand c'est possible peut se formuler en pseudo Algol comme suit

Pas0 : a := 0;

Pas1 : si il existe $i + k.j \in C^{(a)}(G)$, t_1 et t_2 tels que

$$t_1 \neq t_2 \text{ et } \frac{\hat{s}_{t_1}[i + k.j]}{C^{(a)}(G) \cup E(G)} = \frac{\hat{s}_{t_2}[i + k.j]}{C^{(a)}(G) \cup E(G)}$$

alors

début

$$CX^{(a)}(G, m, q) := CX^{(a)}(G, m, q) - \left\{ \frac{\hat{s}_{t_2}[i + k.j]}{C^{(a)}(G) \cup E(G)} \right\};$$

allera Pas1

fin

sinon

si il existe $i + k.j$ et $i' + k'.j' \in C^{(a)}(G)$ tels que

$$i + k.j \neq i' + k'.j' \text{ et } \frac{S[i + k.j]}{C^{(a)}(G) \cup E(G)} \cap \frac{S[i' + k'.j']}{C^{(a)}(G) \cup E(G)} \neq \emptyset$$

et $i + k.j \mathcal{R}_1 i' + k'.j'$ n'est pas vérifiée alors

allera Pas4

sinon

si $C^{(a)}(G)$ et $CX^{(a)}(G, m, q)$ vérifient la condition $\mathcal{C}$ alors allera Pas 3

sinon allera Pas2;

Pas2 : Construire $C^{(a+1)}(G)$;

Construire $CX^{(a+1)}(G, m, q)$;

commentaire $CX^{(a+1)}(G, m, q) = \frac{CX^{(a)}(G, m, q)}{C^{(a+1)}(G)}$;

a := 1 + a; allera Pas1;

Pas3 : fin favorable;

Pas4 : fin défavorable;

La condition $\mathcal{C}_1$ n'étant qu'un cas particulier de la condition à savoir $\mathcal{C}$ est vérifiée pour a = 0, cet algorithme s'arrêtera immédiatement si une grammaire G vérifie la condition $\mathcal{C}_1$;

Exemple 13

Soit la grammaire G_0 définie dans les exemples 1 et 3; l'algorithme se termine par le Pas3 avec a = 0; l'ensemble $CX^{(0)}(G_0, 1, 1)$ vérifie la condition $\mathcal{C}$ comme la table de l'exemple 7 permet de le voir aisément.

La grammaire G_7, définie dans l'exemple 11, provoque la terminaison de l'algorithme par le Pas4 avec a = 0 car cette grammaire ne vérifie pas la condition $\mathcal{C}$ à cause des éléments 3.1 et 4.1 qui ne sont pas dans la relation $\mathcal{R}_1$ (voir exemple 11). La grammaire G_9, définie dans l'exemple 11, provoque la terminaison de l'algorithme par le Pas3 avec a = 3; on a

$C^{(0)}(G_9)$ = 0.1, 0.2, 0.3, 1.1, 2.1, 2.2, 2.3, 2.4, 3.1, 4.1, 4.2, 4.3, 4.4, 5.1

$C^{(1)}(G_9) = C^{(0)}(G_9) - \{2.1, 4.1\}$

1.1 étant l'élément de $C^{(1)}(G_9) = C^{(0)}(G_9)/\mathcal{R}_1^+$ représentant la classe d'équivalence formée des éléments suivants se $C^0(G_9)$:

1.1, 2.1 et 4.1

de la même façon

$C^{(2)}(G_9) = C^{(1)}(G_9) - \{4.2\}$ avec 2.2 = {2.2, 4.2}

$C^{(3)}(G_9) = C^2(G_9) - \{4.3\}$ avec 2.3 = {2.3, 4.3}

donnons le détail de ce dernier ensemble :

$C^3(G_9)$ = 0.1 = {0.1}, 0.2 = {0.2}, 0.3 = {0.3},
1.1 = {1.1, 2.1, 4.1}, 1.2 = {1.2}, 2.2 = {2.2, 4.2},
2.3 = {2.3, 4.3}, 2.4 = {2.4}, 3.1 = {3.1}, 4.4 = {4.4},
5.1 = {5.1}

enfin

$S[0.1] = \{\hat{s}_1[0.1] = <\}$
$S[0.3] = \{\hat{s}_1[0.3] = >\}$
$S[1.1] = \{\hat{s}_1[1.1] = a$
$S[2.3] = \{\hat{s}_1[2.3] = b$

$S[3.1] = \{\hat{s}_1[3.1] = c >\}$
$S[5.1] = \{\hat{s}_1[5.1] = cb\}$
$S[2.2] = \{\hat{s}_1[2.2] = 1.1\ 4.0\}$
$S[4.4] = \{\hat{s}_1[4.4] = 2.3\ 4.0\}$
$S[0.2] = \{\hat{s}_1[0.2] = 0.1\ 1.0\}$
$S[1.2] = \{\hat{s}_1[1.2] = 1.1\ 1.0\}$
$S[2.4] = \{\hat{s}_1[2.4] = 2.3\ 1.0\}$
$U(0.0) = \{\bar{u}_0(0.0) = 0.1\ 0.2\ 0.3\}$
$U(1.0) = \{\bar{u}_0(1.0) = 1.1\ 1.2,\ \bar{u}_1(1.0) = 1.1\ 2.2\ 2.3\ 2.4,$
$\bar{u}_2(1.0) = 3.1\}$
$U(4.0) = \{\bar{u}_0(4.0) = 1.1\ 2.2\ 2.3\ 4.4,\ \bar{u}_1(4.0) = 5.1\}$

composent l'ensemble $CX^{(3)}(G_9, \leqslant 1, \leqslant 1)$, cette dernière notation signifiant que si $CX^{(3)}(G_9, 1, 1)$ vérifie la condition $\mathfrak{C}$, on a réduit le plus possible la longueur des contextes toute en conservant la vérification de cette condition $\mathfrak{C}$;
donnons dans deux colonnes en regard, les chaînes $\hat{s}_t[i + k.j]$ pour les deux ensembles $CX^{(0)}(G_0, 1, 1)$ et $CX^{(0)}(G_0, \leqslant 1, \leqslant 1)$:

$CX^{(0)}(G_0, 1, 1)$			$CX^{(0)}(G_0, \leqslant 1, \leqslant 1)$		
$i + k.j$	t	$\hat{s}_t[i + k.j]$	$i + k.j$	t'	$\hat{s}_{t'}[i + k.j]$
0.1	1	< +	0.1	1	<
0.1	2	< -			
0.1	3	< .			
0.1	4	< d			
0.3	1	0.2 >	0.3	1	0.2 >
1.3	1	1.2 . >	1.3	1	1.2 .
1.3	2	1.2 . d			
2.1	1	0.1 + .	2.1	1	+
2.1	2	0.1 + d			
3.1	1	0.1 - .	3.1	1	-
3.1	2	0.1 - d			
4.1	1	0.1 . >	4.1	1	0.1 .
4.1	2	0.1 d .	4.1	2	0.1 d
4.1	3	0.1 dd			
4.1	4	0.1 . d			
6.1	1	1.1 . >	6.1	1	1.1 .
6.1	2	1.3 >	6.1	2	1.3 >
6.1	3	1.1 . d			
7.2	1	7.1 d .	7.2	1	7.1 d
7.2	2	7.1 dd			
7.2	3	7.1 d >			

8.1	1	1.1 d .	8.1	1.1 d
8.1	2	1.1 dd	8.1	1.3 d
8.1	3	1.3 d >		
8.1	4	1.3 dd		
5.1	1	1.1 7.0 .	5.1	7.0 .
5.1	2	1.3 7.0	5.1	7.0 >
7.1	1	1.1 7.0 d	7.1	7.0 d
7.1	2	1.3 7.0 d		
1.2	1	1.1 5.0 .	1.2	1.1 5.0
1.4	1	1.3 5.0 >	1.4	1.3 5.0
1.1	1	0.1 2.0 .	1.1	2.0
1.1	2	0.1 2.0 d		
0.2	1	0.1 1.0 >	0.2	1.0

montrons que cet algorithme(*) s'arrête toujours après un nombre fini de pas élémentaries; supposons que cet algorithme ait effectué a itérations; il y a alors trois possibilités au total :

1) $C^{(a)}(G)$ et $CX^{(a)}(G, m, k)$ vérifient $\mathfrak{C}$ auquel cas l'algorithme est terminé (fin par Pas3)

2) $C^{(a)}(G)$ et $CX^{(a)}(G, m, k)$ ne vérifient pas $\mathfrak{C}$ et il existe $i + k.j$ et $i' + k'.j' \in C^{(a)}$§G) tels que
$\frac{S[i + k.j]}{C^{(a)}(G) \cup E(G)} \cap \frac{S[i' + k'.j']}{C^{(a)}(G) \cup E(G)} \neq \emptyset$ et $i + k.j \neq i' + k'.j'$
et $i + k.j \; \mathfrak{R}_1 \; i' + k'.j'$ n'est pas vérifiée auquel cas l'algorithme est terminé (fin par Pas4).

3) ni 1) ni 2) ne sont réalisés ce qui signifie exactement :
il existe $i + k.j$ et $i' + k'.j' \in C^{(a)}(G)$ tels que
$i + k.j \; \mathfrak{R}_1 \; i' + k'.j'$ et $i + k.j \neq i' + k'.j'$;
l'exécution du Pas2 entraînera alors
$Card(C^{(a+1)}(G)) < Card(C^{(a)}(G))$ puisque les éléments $i + k.j$ et $i' + k'.j'$ auront été rangés dans la même classe de $C^{(a+1)}(G) = C^{(a)}(G)/\mathfrak{R}_1^+$
Ces considérations montrent que le nombre d'itérations de cet algorithme est borné supérieurement par Card(C(G)) qui est un nombre fini;
nous employerons désormais les notations $C^{(a)}(G)$ et $CX^{(a)}(G, m, q)$ en sous entendant que ces deux ensembles satisfont la condition $\mathfrak{C}$;
on peut alors donner le résultat suivant :

(*) Les premières instructions après l'étiquette Pas1 réalisent la suppression de n-1 exemplaires de la même chaîne présente à n exemplaires dans $CX^{(a)}(G, m, q)$

pour $\forall \alpha \in W = C(G)/R_{symb}$ et tout $i + k.j \in C^{(a)}(G)$,
on ne peut avoir qu'une des deux relations suivantes :
$i + k.j \subseteq \alpha$ ou bien $i + k.j \cap \alpha = \emptyset$

ce qui peut s'exprimer de la façon suivante :
tous les éléments $c + e.d$ de $C(G)$ qui appartiennent à une même classe $i + k.j$ de $C^{(a)}(G)$ appartiennent aussi tous à une même classe α de W;
en effet, supposons le contraire :
soit $c + e.d$ et $c' + e'.d' \in C(G)$ avec $symb[c + e.d] \neq symb[c' + e'.d']$; alors $c + e.d \, \mathfrak{R}_1 \, c' + e'.d'$ entraîne qu'il existe t et t' tels que :

$$\hat{s}_t[c + e.d] = \hat{s}_{t'}[c' + e'.d'], \; d < n(c + e) \text{ et } d' < n(c' + e')$$

d'où, au moins :

$$cl[c + e.d] = cl[c' + e'.d'], \; d < n(c + e), \; d' < n(c' + e');$$

si on se rapporte à la définition page 26, ceci est équivalent à $symb\,c[+ e.d] = symb[c' + e'.d']$ car le cas où $symb[c + e.d] = \varepsilon$ ou $symb[c' + e'.d'] \varepsilon$est exclus car il entraîne $d = 1 = n(c + e)$ ou $d' = 1 = n(c' + e')$ contrairement à ce qui est supposé; il reste que la conclusion est contraire à l'hypothèse de départ ce qui démontre le résultat cherché; (Note)

Exemple 14

Les éléments de $C(G_9)/R_{symb}$ étant :

$C(G_9)/R_{symb} = \{ <= \{0.1\}, > = \{0.3\},$
$a = \{1.1, 2.1, 4.1\}, b = \{2.3, 4.3\}, c = \{3.1, 5.1\},$
$S = \{0.2, 1.2\}, T = \{2.2, 4.2, 4.4\}\};$

Note : Cette propriété entraîne que la fonction symb est définie de façon évidente de $C^{(a)}(G)$ sur W.

on peut vérifier, d'après l'exemple 13 que la propriété est satisfaite entre les éléments de $C^{(3)}(G_9)$ et $C(G_9)/R_{symb}$, à savoir

$< = 0.1$ $\quad\quad$ $S = 0.2 \cup 1.2$

$> = 0.3$ $\quad\quad$ $T = 2.2 \cup 4.4$

$a = 1.1$

$b = 2.3$

$c = 3.1 \quad 5.1$

on peut voir que la fonction symb conserve tout son sens quand elle est définie comme suit :

$$symb : C^{(3)}(G_9) \rightarrow W_g$$

un autre résultat est :

Tout $i + k.n(i + k) \in C(G)$ est un élément de $C^{(a)}(G)$

Ceci est évident d'après la définition de $\mathcal{C}_1$; on peut déduire de ces quelques propriétés des ensembles $C^{(a)}(G)$ et $CX^{(a)}(G, m, q)$ que l'analyseur syntaxique qui sera construit en utilisant comme vocabulaire de pile $C^{(a)}(G) \cup E$ et comme chaînes de contexte, les chaînes de $CX^{(a)}(G, m, q)$, sera déterministe; de plus, on saura toujours quand opérer une réduction d'après la dernière propriété.

Exemple 15

Considérons la grammaire $G_{,}$; on voit que les éléments suivants font partie de $C^{(3)}(G\ )$:

$0.3 = \{0.3\}$; $1.2 = \{1.2\}$; $2.4 = \{2.4\}$; $3.1 = \{3.1\}$ $4.4 = \{4.4\}$;

$5.1 = \{5.1\}$;

c'est-à-dire tous les éléments de la forme $i + k.n(i + k)$ forment une classe d'équivalence réduite à un seul élément de $C^{(0)}(G_9)$; supposons que $2.4 \quad C^{(3)}(G_9)$ soit sur le sommet de la pile d'un analyseur; ceci

entraîne que la chaîne lisible sur le sommet de la pile est $\overline{u}_1(1.0)$ = 1.1 2.2 2.3 2.4 (cf. exemple 13); on en déduit qu'on doit effectuer la réduction de 2.1 2.2 2.3 2.4 en 1.0; de même, quand 4.4 est sur le sommet de la pile, la chaîne $\overline{u}_0(4.0)$ = 1.1 2.2 2.3 4.4 est lisible sur le sommet de la pile; on doit donc réduire 4.1 4.2 4.3 4.4 en 4.0.

Une grammaire G pour laquelle un tel analyseur peut être construit, sera dite être COORD(m, q).

Nous avons donc atteint le but fixé par les demandes 1 et 2 de l'introduction par des moyens relativement simples (par rapport à la méthode LR(k) par exemple). Il reste à voir la construction effective de l'analyseur syntaxique et à comparer la généralité de notre méthode syntaxique par rapport à celle d'autres déjà connues.

CHAPITRE 5

L'ANALYSEUR DETERMINISTE

C'est un automate à pile

APD(G) = (Q, I, E, P);

Q est l'ensemble des états : q_I initial, q_C courant, q_F final;
E est le ruban d'entrée, d'alphabet T;
P est la pile d'alphabet $C^{(a)}(G) \cup E(G)$;
I est un ensemble d'instructions représentées par des quintuplets (ℓc, rc, co, rd, sh) avec :
ℓc est une chaîne sur l'alphabet de P, de longueur $\leq m + 1$;
rc est une chaîne sur l'alphabet de E, de longueur $\leq q + 1$;
co est un élément $i + k.j \in C^{(a)}(G)$;
rd est une variable booléenne;
sh est une variable booléenne;
désignons par (P) la chaîne de symboles contenue dans P, son sommet étant considéré à droite et par (E) la partie droite de la phrase terminale non encore lue par l'automate; l'interprétation d'un quintuplet par l'automate APD(G) est alors le suivant

si(P) : |ℓc| = ℓc et |rc| : (E) = rc alors
  début
  si ℓc : 1 ∈ E(G) alors effacer(P) : 1 de P;
  écrire co sur P;
  si rd alors
    début
    effacer(P) : j de P;
    commentaire co = i + k.j avec j = n(i + k);
    écrire i.0 sur P;
    fin;
 si sh alors faire progresser E d'un symbole;
 fin;

les quintuplets sont formés à partir de $CX^{(a)}(G, m, q)$ comme suit :
pour toute chaîne $\hat{s}_t[i + k.j]$, on crée le quintuplet (ℓc, rc, co, rd, sh) avec :

si symb|i + k.j| ∈ N alors
  ℓc = $\bar{\varphi}_t[i + k.j]$ cℓ[i + k.j] et rc = symb $\bar{z}_t[i + k.j]$;
sinon
  ℓc = $\bar{\varphi}_t[i + k.j]$ et rc = cℓ[i + k.j] symb $\bar{z}_t[i + k.j]$;

co = i + k.j;
si j = n(i + k) alors rd = vrai sinon rd = faux;
si cℓ[i + k.j] ∈ E(G) ou symb[i + k.j] = ε alors sh = faux
sinon sh = vrai;
l'ensemble de ces quintuplets constitue une table de décision pour l'analyseur APD(G); la façon dont elle est construite à partir de l'ensemble $CX^{(a)}(G, m, q)$ vérifiant la condition de déterminisme $\mathcal{C}$ entraîne qu'il n'y a pas deux quintuplets égaux dans une telle table; l'état initial q_I correspond aux quintuplets tels que 1 : rc = ⊢ ; l'état final q_F correspond au quintuplet ℓc : 1 = 0.3; l'état courant q_C correspond à tous les autres quintuplets; d'un point de vue pratique les tables de décision se présenteront sous la forme de quintuplets (ℓc, rc, co, a_1, a_2) déduits de (ℓc, rc, co, rd, sh) de la façon suivante :
a_1 = si rd alors 'Réduire i.0' sinon' '; (REDUCE i.0)
a_2 = si sh alors ' ' sinon 'ne pas avancer E' (NO SHIFT)
de plus, l'état final q_F correspondra au quintuplet où a_1 = 'Reduire 0.0';
une autre différence, au niveau de l'application, est que la longueur des contextes n'est pas uniforme dans la table de décision; en effet, m et q sont des bornes supérieures pour les longueurs des contextes : les longueurs des contextes ne sont uniformes que pour les éléments de la table de décision tels que cℓ[i + k.j] soit le même; ces longueurs sont minimales et telles que la condition de déterminisme soit conservée; il en résulte une diminution importante de la taille d'une table de décision pour une grammaire donnée; par contre les erreurs syntaxiques seront parfois détectées avec retard; cette question classique est discutée dans [KNU 65] page 621 ainsi que dans [KOR 69] page 623.

Exemple 16

On considère la CF grammaire G = (N, T, R, Z) [KNU 65] p. 619
avec N = {Z, S};
T = {⊢, ⊣, (,), a, +};
R = {Z → ⊢ S ⊣
S → (S + S) | a};
L(G) est un langage d'expressions complètement parenthésées;
pour toute phrase x telle que ⊢ x ⊣ ∈ L(G), on se livre aux opérations de "sabotage" suivantes :

1) supprimer le plus grand facteur gauche de x ne contenant que des parenthèses ouvrantes; soit x_1 le résultat obtenu

2) supprimer le plus grand facteur droit de x_1 ne contenant que des parenthèses fermantes; soit x_2 le résultat obtenu;

3) supprimer tous les signes + apparaissant dans x_2; soit x_3 le résultat obtenu;

4) remplacer toute occurrence d'une parenthèse dans x_3 par une occurrence du symbole b; soit y le résultat obtenu;

On a ainsi défini une application T : L(G) → T(L(G))
avec T(L(G) ⊂ ⊢ {a, b}* ⊣ qui a toute phrase ⊢x⊣ de L(G) fait correspondre un mot ⊢y⊣ = T(⊢ x⊣).

Le problème posé est : T est-elle une application injective autrement dit l'opération de "sabotage" est-elle toujours réversible ?

On va d'abord écrire une grammaire G'_6 telle que $L(G'_6) = L(G)$ et telle qu'on puisse lui associer facilement une grammaire G_6 avec $L(G_6) = T(L(G)) = T(L(G'_6))$.

Les grammaires G'_6 et G_6 sont maintenant définies :

avec :

$N'_6 = \{Z, B, L, R, N\}$;
$T'_6 = \{\vdash, \dashv, (,), a, +\}$;
$R'_6 = \{$

$Z_{0.0} \rightarrow \vdash_{0.1} B_{0.2} \dashv_{0.3}$
$B_{1.0} \rightarrow a_{1.1} \mid (_{2.1} L_{2.2} +_{2.3} R_{2.4})_{2.5}$
$L_{3.0} \rightarrow a_{3.1} \mid (_{4.1} L_{4.2} +_{4.3} N_{4.4})_{4.5}$
$R_{5.0} \rightarrow a_{5.1} \mid (_{6.1} N_{6.2} +_{6.3} R_{6.4})_{6.5}$
$N_{7.0} \rightarrow a_{7.1} \mid (_{8.1} N_{8.2} +_{8.3} N_{8.4})_{8.5}$
};

avec :

$N_6 = N'_6$;
$T_6 = \{\vdash, \dashv, a, b\}$;
$R_6 = \{$

$Z_{0.0} \rightarrow \vdash_{0.1} B_{0.2} \dashv_{0.3}$
$B_{1.0} \rightarrow a_{1.1} \mid L_{2.1} R_{2.2}$
$L_{3.0} \rightarrow a_{3.1} \mid L_{4.1} N_{4.2} b_{.43}$
$R_{5.0} \rightarrow a_{5.1} \mid b_{6.1} N_{6.2} R_{6.3}$
$N_{7.0} \rightarrow a_{7.1} \mid b_{8.1} N_{8.2} N_{8.3} b_{8.4}$
};

On remarquera que G'_6 et G_6 ont les mêmes symboles auxiliaires sujets de règles "analogues" ayant même nombre de productions qui mettent en évidence les différentes opérations de T.

La solution conditionnelle du problème se formule alors ainsi : si pour les mots de $T(L(G)) = L(G_6)$, il existe un analyseur déterministe produit à partir de G_6 (ce qui prouve que G_6 est non ambiguë) alors on peut construire un transducteur de T(L(G)) en L(G) à partir des grammaires G_6 et G'_6.

Or G_6 est une grammaire COORD(1, 1) dont la table de décision construite à partir de $CX^{(2)}(G_6, \leqslant 1, \leqslant 1)$ est :

numéro d'instruction	ℓc	rc	co	a_1	a_2
1	ε	⊢	0.1		
2	ε	⊣	0.3		Réduire 0.0
3	0.1	a⊣	1.1		Réduire 1.0
4	0.1	aa	3.1		Réduire 3.0
5	0.1	ab	3.1		Réduire 3.0
6	2.1	a	5.1		Réduire 5.0
7	6.2	a	5.1		Réduire 5.0
8	2.1	ab	7.1		Réduire 7.0
9	6.1	aa	7.1		Réduire 7.0
10	6.2	ab	7.1		Réduire 7.0
11	6.1	ab	7.1		Réduire 7.0
12	4.1	b	4.3		Réduire 3.0
13	2.1	b	6.1		
14	6.2	b	6.1		
15	6.1	b	6.1		
16	8.3	b	8.4		Réduire 7.0
17	2.1 7.0	ε	4.2	ne pas avancer E	
18	6.1 7.0	ε	6.2	ne pas avancer E	
19	6.2 7.0	ε	8.3	ne pas avancer E	
20	2.1 5.0	ε	2.2	ne pas avancer E	
21	6.2 5.0	ε	6.3	ne pas avancer E	
22	3.0	ε	2.1	ne pas avancer E	
23	1.0	ε	0.2	ne pas avancer E	

donc G_6 est non-ambiguë;

Soit la phrase ⊢x⊣ de L(G) :

⊢((a + a) + (a + (a + a)))⊣

On a :

⊢y⊣ = T(⊢x⊣) = ⊢aabbabaa⊣

La liste des descriptions instantanées de l'analyseur APD(G_6) pour la phrase ⊢y⊣ apparaît dans la table suivante :

P	E	ligne de la table de décision	numéro de production
	aabbabaa	1	
0.1	aabbabaa	4	
0.1 3.1	abbabaa		3
0.1 3.0	abbabaa	22	
0.1 2.1	abbabaa	8	
0.1 2.1 7.1	bbabaa		7
0.1 2.1 7.0	bbabaa	17	
0.1 2.1 4.2	bbabaa	12	
0.1 2.1 4.2 4.3	babaa		4
0.1 3.0	babaa	22	
0.1 2.1	babaa	13	
0.1 2.1 6.1	abaa	11	
0.1 2.1 6.1 7.1	baa		7
0.1 2.1 6.1 7.0	baa	18	
0.1 2.1 6.1 6.2	baa	14	
0.1 2.1 6.1 6.2 6.1	aa	9	
0.1 2.1 6.1 6.2 6.1 7.1	a		7
0.1 2.1 6.1 6.2 6.1 7.0	a	18	
0.1 2.1 6.1 6.2 6.1 6.2	a	7	
0.1 2.1 6.1 6.2 6.1 6.2 5.1			5
0.1 2.1 6.1 6.2 6.1 6.2 5.0		21	
0.1 2.1 6.1 6.2 6.1 6.2 6.3			6
0.1 2.1 6.1 6.2 5.0		2.1	
0.1 2.1 6.1 6.2 6.3			6
0.1 2.1 5.0		20	
0.1 2.1 2.2			2
0.1 1.0		23	
0.1 0.2		2	
0.1 0.2 0.3			0
0.0			

On possède maintenant, à partir de la liste des numéros de production obtenus à une génération en utilisant la grammaire G'_6

numéro de production	phrase obtenue
	Z
0	⊢ B ⊣
2	⊢ (L + R) ⊣
6	⊢ (L + (N + R)) ⊣
6	⊢ (L + (N + (N + R))) ⊣
5	⊢ (L + (N + (N + a))) ⊣
7	⊢ (L + (N + (a + a))) ⊣
7	⊢ (L + (a + (a + a))) ⊣
4	⊢ ((L + N) + (a + (a + a))) ⊣
7	⊢ ((L + a) + (a + (a + a))) ⊣
3	⊢ ((a + a) + (a + (a + a))) ⊣

ce qui permet de retrouver la phrase ⊢ x ⊣ dont on doit parti.
Le couple (analyseur selon G_6, générateur selon G'_6) est bien le transducteur recherché (cf. [ALP 71]).

CHAPITRE 6

COMPARAISON AVEC D'AUTRES SOUS-ENSEMBLES DES C-F GRAMMAIRES

Les sous-ensembles auxquels nous nous intéresserons sont :

a) les grammaires LL(k)

b) les grammaires BRC(n, k) (de contexte borné)

c) les grammaires LR(k)

1) rappels sur les grammaires LL(k)

La méthode LL(k) est une méthode descendante déterministe; l'analyseur LL(k) comporte un générateur à pile(*) procédant par applications successives de la relation $\xrightarrow{L}$ à partir de l'axiome de la grammaire "en direction" de la phase terminale à analyser; supposons qu'on ait à analyser une phrase $x \in L(G)$; soit x_1 la partie déjà lue de x et x_2 la partie encore à lire c'est-à-dire que $x = x_1 x_2$; le générateur contient sur sa pile la chaîne $x_1 \varphi$ telle que $Z \xrightarrow{L*} x_1 \varphi$; à ce point de l'analyse, il y a deux possibilités :

a) $k : \varphi \in T^*$ alors si $x \in L(G)$ on doit avoir
$k : \varphi = k : x_2$
après constat de cette égalité, le ruban d'entrée contenant x avance d'un symbole et on passe au pas suivant de l'analyse.

b) $k : \varphi \notin T^*$ c'est-à-dire contient au moins un symbole auxiliaire;
supposons que $k : \varphi = yA\Psi$; $A_{f.o}$ est le sujet des productions φ_{f+h} avec $0 \leq h < p(A_{f.o})$;
le générateur a alors le choix de générer les $p(A_{f.o})$ chaînes
$x_1 y \varphi_{f+h} \Psi \quad : (|\varphi| - k)$;
or on veut que ce générateur soit déterministe c'est-à-dire trouver un unique k_o tel que
$k : x_2 = k : (y \varphi_{f+h_o} \Psi \varphi : (|\varphi| - k))$

* l'alphabet de pile est V

ceci entraîne évidemment la condition sur les productions de $A_{f.o}$:

$$h_1 \neq h_2 \Leftrightarrow H_{k'}(\varphi_{f+h_1}\,\psi\varphi\colon (|\varphi| - k)) \cap H_{k'}(\varphi_{f+h_2}\,\psi\varphi\colon (|\varphi| - k)) = \emptyset$$

pour $\forall h_1$ et h_2 et $1 \leq k' \leq k$ ces valeurs de k' étant justifiées par le fait que $0 \leq |y| \leq k$; si cette condition est vérifiée, le générateur produit donc la chaîne $x_1 y\,\varphi_{f+h}\,\psi\varphi\colon (|\varphi| - k)$, le ruban d'entrée contenant x avance d'une position et on passe au pas suivant de l'analyse; on déduit de cette explication, deux conditions sur les grammaires LL(k) :

Condition 1 : pour qu'une CF grammaire G soit LL(k), il est nécessaire que la relation

$A \overset{L+}{\Rightarrow} A\theta$ ne soit vérifiée pour aucun $A \in N$

Condition 2 : pour qu'une CF grammaire G soit LL(k), il est nécessaire que pour $\forall A_{f.o} \in N$ de productions φ_{f+h} avec $0 \leq k \leq p(A_{f.o})$

$$h_1 \neq h_2 \Leftrightarrow H_k(\varphi_{f+h_1}\varphi) \cap H_k(\varphi_{f+h_2}\varphi) = \emptyset$$
$$\text{pour } \forall \varphi \text{ tel que } Z \overset{L+}{\Rightarrow} xA\varphi$$

la satisfaction de ces deux conditions par une CF grammaire G est une condition nécessaire et suffisante pour qu'elle soit LL(k);
d'après le fonctionnement déterministe de l'analyseur on peut aussi formuler ces conditions de façon différente :

Condition 3 : une condition nécessaire et suffisante pour qu'une CF grammaire G soit LL(k) est que pour $\forall x_1$, A, φ_1, φ_1', h_1, h_2, x_2, x_2' tels que

$$Z \overset{L+}{\Rightarrow} x_1 A\varphi_1 \overset{L}{\Rightarrow} x_1 \varphi_{f+h_1}\,\varphi_1 \overset{L*}{\Rightarrow} x_1 x_2$$
$$\text{et } Z \overset{L+}{\Rightarrow} x_1 A\varphi_1' \overset{L}{\Rightarrow} x_1 \varphi_{f+h_2}\,\varphi_1' \overset{L*}{\Rightarrow} x_1 x_2'$$

l'égalité $k : x_2 = k : x_2'$ entraîne $h_1 = h_2$
(cf [KNU 67]);

Exemple 17

La grammaire G_o définie dans les exemples 1 et 3 n'est pas LL(k) quelque soit k car elle a une règle récursive à gauche :

$$N_{7.0} \rightarrow N_{7.1}\ d_{7.2} \mid d_{8.1}$$

La grammaire G_9 définie dans l'exemple 11 n'est pas LL(k) quelque soit k à cause de la règle

$$S_{1.0} \rightarrow a_{1.1}\ S_{1.2} \mid a_{2.1}\ T_{2.2}\ b_{2.3}\ S_{2.4} \mid c_{3.1}$$

en effet :

$$H_k(a_{1.1}\ S_{1.2}\ \varphi) \cap H_k(a_{2.1}\ T_{2.2}\ b_{2.3}\ S_{2.4}\ \varphi) = a^k$$

quels que soient k, x, φ tels que $Z \overset{L+}{\rightarrow} xS\varphi$.

La grammaire G_7 définie dans l'exemple 11 est LL(k) pour $k \geq 3$ pour la règle $S_{1.0} \rightarrow x_{1.1}\ A_{1.2}\ a_{1.3} \mid y_{2.1}\ A_{2.2}$ on a

$H_1(x_{1.1}\ A_{2.2}\ a_{1.3}) = x$

$H_1(y_{2.1}\ A_{2.2}) = y$

et pour la règle $A_{3.0} \rightarrow a_{3.1}\ A_{3.2} \mid a_{4.1}$ on a

$H_3(a_{3.1}\ A_{3.2}\ a_{1.3}\ >_{0.3}) = aaa$

$H_2(a_{4.1}\ a_{1.3}\ >_{0.3}) = aa>$

$H_2(a_{3.1}\ A_{3.2}\ a_{1.3}\ >_{0.3}) = aa$

$H_2(a_{4.1}\ >_{0.3}) = a\ >$

2) rappels sur les grammaires BRC(n, k)

La méthode BRC(n, k) est une méthode ascendante déterministe; l'analyseur BRC(n, k) comporte un reconnaisseur à pile(*) procédant par applications successives de la relation $\overset{R}{\rightarrow}$ à partir de la phrase terminale à analyser "en direction" de l'axiome de la grammaire; supposons

* l'alphabet de pile est V

qu'on ait à analyser une phrase $x \in L(G)$; soit x_1 la partie déjà lue de x et qui a été réduite en φ_1 et x_2 la partie encore à lire c'est-à-dire que $x = x_1x_2$; la pile contient φ_1 telle que $Z \overset{R*}{\Rightarrow} \varphi_1 x_2$;
à ce point de l'analyse il y a deux possibilités dûes à l'examen de la chaîne $\varphi_1 : n\ k : x_2$

a) on trouve que $\varphi_1 : 1$ est le dernier symbole d'une production; on remplace alors cette production par son sujet sur lapile et on passe au pas suivant de l'analyse (on a effectué une réduction).

b) $\varphi : 1$ n'est pas le dernier symbole d'une production; on écrit $1 : x_2$ sur la pile, on avance le ruban d'entrée d'un symbole et on passe au pas suivant de l'analyse (on a effectué un "shift").

On déduit de cette explication, une condition nécessaire et suffisante après avoir introduit la notation suivante :

à partir de l'ensemble de chaînes attaché au symbole $i + k.j \in C(G)$

$$\{\overline{\sigma}_t[i + k.j] \mid 1 \leq t \leq \overline{ns}[i + k.j]\}$$

défini page 24 on forme l'ensemble

$$L[i + k.j] = \{symb[\overline{\sigma}_{t'}, [i + k.j]] \mid 1 \leq t' \leq ns[i + k.j] \leq \overline{ns}[i + k.j]\}$$

La condition nécessaire et suffisante pour qu'une CF grammaire G soit BRC(n, k) est que
pour $\forall \alpha \in V$ on ait

$$\Big(\bigcup_{\substack{i + k.j \in \alpha \\ \text{avec} \\ j < n(i + k)}} L[i + k.j]\Big) \cap \Big(\bigcup_{i + k.n(i + k) \in \alpha} L[i + k.n(i + k)]\Big) = \emptyset$$

et
$$\bigcup_{\substack{i_1 + k_1.n(i_1 + k_1) \in \alpha \\ i_2 + k_2.n(i_2 + k_2) \in \alpha \\ \text{et } i_1 + k_1 \neq i_2 + k_2}} (L[i_1 + k_1.n(i_1 + k_1)] \cap L[i_2 + k_2.n(i_2 + k_2)]) = \emptyset$$

de la même façon que précédemment, on peut donner une autre condition nécessaire et suffisante pour qu'une CF grammaire G soit BRC(n, k) pour $\forall \varphi, \Psi$, x, x' tels que :

$$Z \overset{R*}{\rightarrow} \varphi x$$

et $$Z \overset{R*}{\rightarrow} \varphi' x'$$

l'égalité φ: n k : x = φ' : n k : x' entraîne l'une des deux possibilités suivantes :

a) φ: 1 = symb[c + e.d] avec d < n(c + e) et φ' : 1 = symb[f + h.g] avec g < n(f + h); on a c + e.d ≡ f + h.g (mod. R_{symb}).

b) φ: 1 = φ' : 1 = i + k.n(i + k)

Exemple 18

Soit la grammaire $G_1 = (N_1, T_1, R_1, Z)$ avec

$N_1 = \{Z, S, A\}$, $T_1 = \{<, >, a, b, c\}$

$R_1 = \{Z_{0.0} \rightarrow <_{0.1} S_{0.2} >_{0.3}$

$S_{1.0} \rightarrow a_{1.1} A_{1.2} c_{1.3} \mid b_{2.1}$

$A_{3.0} \rightarrow a_{3.1} S_{3.2} c_{3.3} \mid b_{4.1}\}$

G_1 n'est pas BRC(n, k) quels que soient n et k bornés parce que

$L[2.1] \cap L[4.1] = a^n b^{k+1}$

La grammaire G_o définie dans les exemples 1 et 3 est BRC(1, 1); il suffit pour s'en assurer de former les ensembles de chaînes L[i + k.j] à partir des chaînes $\overline{\sigma}_t[i + k.j]$ de l'exemple 7; on obtient :

t'	symb $\sigma_{t'}$	$[i+k.j]$ / $i + k.j$
1	< +	0.1
2	< -	0.1
3	< .	0.1
4	< d	0.1
1	R >	0.3
1	M . >	1.3
2	M . d	1.3
1	< + .	2.1
2	< + d	2.1
1	< - .	3.1
2	< - d	3.1
1	< . >	4.1
2	< d .	4.1
3	< d d	4.1
4	< . d	4.1
1	S . >	6.1
2	. >	6.1
3	S . d	6.1
1	N d .	7.2
2	N d d	7.2
3	N d >	7.2
1	D d .	8.1
2	S d d	8.1
3	. d >	8.1
4	. d d	8.1
1	S N .	5.1
2	. N >	5.1
1	S N d	7.1
2	. N d	7.1
1	S M .	1.2
1	. M >	1.4
1	< S .	1.1
2	< S d	1.1
1	< R >	0.2

et on voit que la condition donnée page 32 est vérifiée.

3) rappels sur les grammaires LR(k)

La méthode LR(k) est une méthode ascendante déterministe; l'analyseur LR(k) comporte un reconnaisseur à pile procédant par applications successives de la relation $\overset{R}{\rightarrow}$ à partir de la phrase terminale à analyser "en direction" de l'axiome de la grammaire; la pile est divisée en deux pistes parallèles; l'une de vocabulaire V joue exactement le même rôle que dans un analyseur BRC(n, k) alors que l'autre dont le vocabulaire comprend $C(G) \cup E(G)$ permet de connaître l'état de l'analyseur pour le symbole en regard sur la première piste.

Sans entrer plus avant dans l'explication du fonctionnement d'un analyseur LR(k) (voir [COU 68]), nous pouvons indiquer une condition nécessaire et suffisante pour qu'une CF grammaire G soit LR(k) à savoir :

pour $\forall \varphi, \varphi', x, x'$ tels que

$$Z \overset{R*}{\rightarrow} \varphi x$$

$$Z \overset{R*}{\rightarrow} \varphi' x'$$

l'égalité $\varphi k : x = \varphi' k : x'$ entraîne l'une des deux possibilités suivantes :

a) $\varphi : 1 = \text{symb}[c + e.d]$ avec $d < n(c + e)$ et
$\varphi' : 1 = \text{symb}[f + h.g]$ avec $g < n(f + h)$

b) $\varphi : 1 = \varphi' : 1 = \text{symb}[i + k.n(i + k)]$

$(c + e.d, f + h.g, i + k.n(i + k)$ sont des éléments de $C(G))$

Exemple 19

Soit la grammaire G_6 décrite dans l'exemple 16; elle est LR(1) comme on peut le voir dans [KNU 65] p. 620 ou dans [COU 68] p. 30 à 35.

Soit la grammaire $G_8 = (N_8, T_8, R_8, Z)$ avec

$N_8 = \{Z, IFST, E\}$, $T_8 = \{<, >, \text{if}, (,), =, ;, \text{then}, a\}$

et $R_8 = \{Z_{0.0} \rightarrow {<}_{0.1}\ IFST_{0.2}\ {>}_{0.3}$

$$IFST_{1.0} \rightarrow \text{if}_{1.1}\ (_{1.2}\ E_{1.3}\ )_{1.4}\ =_{1.5}\ E_{1.6}\ ;_{1.7} \mid$$

$$\text{if}_{2.1}\ E_{2.2}\ =_{2.3}\ E_{2.4}\ \text{then}_{2.5}$$

$$E_{3.0} \rightarrow (_{3.1}\ E_{3.2}\ )_{3.3} \mid a_{4.1}\}$$

Elle n'est pas LR(k) quelque soit k car

$$Z \overset{R*}{\rightarrow} < \text{if(E)}\ x_1 = \text{symb}[0.1\ 1.1\ 1.2\ 1.3\ 1.4]\ x_1$$

$$Z \overset{R*}{\rightarrow} < \text{if(E)}\ x_2 = \text{symb}[0.1\ 2.1\ 3.1\ 3.2\ 3.3]\ x_2$$

avec = E ; $\overset{*}{\rightarrow} x_1 \in T_7^*$

et = E then $\overset{*}{\rightarrow} x_2 \in T_7^*$

il est évident que quelque soit k on a toujours

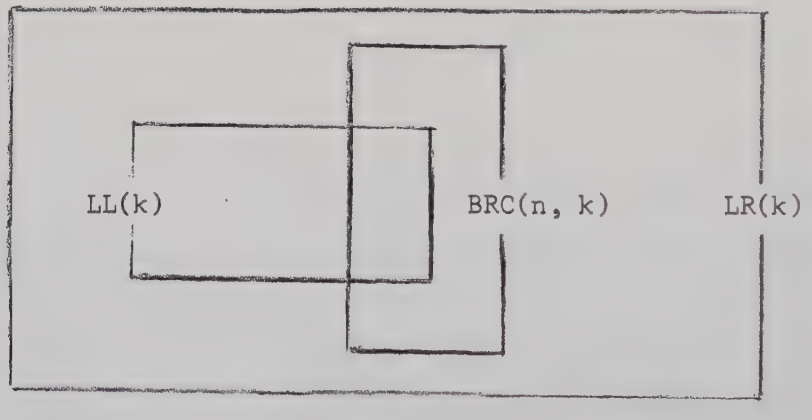
$$H_k(= \text{E ;}) \cap H_k(= \text{E then}) \neq \emptyset;$$

par exemple la chaîne :

$= (^{k-1}$

appartient à l'intersection de ces deux ensembles;

Ces rappels étant faits, nous rappellons également les relations entre ces ensembles sous la forme d'un schéma d'inclusion



Résultat 1 Il existe des CF grammaires qui sont LL(k) (et par suite LR(k)) et qui ne sont pas COORD(n, q) quels que soient n et q bornés;

On peut exhiber de telles grammaires, par exemple comme suit : soit une règle visiblement LL(1)

$$A \rightarrow a\varphi_1 \mid b\varphi_2$$

faisons-en sorte que $\varphi_1 \overset{*}{\Rightarrow} \{c\}^*$ et $\varphi_2 \overset{*}{\Rightarrow} \{c\}^*$ par deux séquences de dérivations différentes, par exemple

$\varphi_1 = Bccc$ et $\varphi_2 = Bc$ avec $B \overset{*}{\Rightarrow} \{c\}^*$

introduisons pour B une règle récursive à droite (i.e provoquant un allongement de la pile dans un analyseur ascendant)

$B \rightarrow cB \mid c$

nous obtenons au total :

$A_{1.0} \rightarrow a_{1.1}\ B_{1.2}\ c_{1.3}\ c_{1.4}\ c_{1.5} \mid b_{2.1}\ B_{2.2}\ c_{2.3}$

$B_{3.0} \rightarrow c_{3.1}\ B_{3.2} \mid c_{4.1}$

grammaire qui est LL(5); par contre nous voyons que
$S[3.1] \cap S[4.1] = 3.1^n\ ccc>$ quels que soient n et q; (on notera que si la règle de sujet B est écrite récursive à gauche, il n'y a plus aucun problème).

Résultat 2 Il existe des CF grammaires qui sont COORD(n, q) et qui ne sont pas LL(k) quelque soit k borné.

Pour en exhiber des exemples, on peut bien sûr considérer les grammaires qui sont COORD(n,q) et qui, pour au moins un symbole auxiliaire A, sont telles que : $A \overset{L+}{\Rightarrow} A\varphi$;
on peut également considérer une règle telle que :

$A \rightarrow \varphi_1\ a \mid \varphi_2\ b$

faisons en sorte que $\varphi_1 \overset{*}{\Rightarrow} \{c\}^*$ et $\varphi_2 \overset{*}{\Rightarrow} \{c\}^*$ si bien que
$H_k(\varphi_1\ a) \cap H_k(\varphi_2\ b) \ni c^k$ pour $\forall k$; le plus simple est $\varphi_1 = \varphi_2 = B$
et $B \rightarrow c\ B \mid c$;
on obtient donc

$A_{1.0} \rightarrow B_{1.1}\ a_{1.2} \mid B_{2.1}\ b_{2.2}$

$B_{3.0} \rightarrow c_{3.1}\ B_{3.2} \mid c_{4.1}$

grammaire qui est COORD(1, 1)

Résultat 3 Il existe des CF grammaires qui sont LR(k) et qui ne sont ni LL(k') ni COORD(n, q) quels que soient k', n, q bornés

il suffit pour en exhiber des exemples de construire un exemple comme suit : on reprend l'exemple donné pour le résultat 1 et on s'arrange pour qu'il ne soit pas LL(k') quelque soit k' :

$A_{1.0} \rightarrow X_{1.1}\ B_{1.2}\ c_{1.3}\ c_{1.4}\ c_{1.5} \mid Y_{2.1}\ B_{2.2}\ c_{2.3}$

$B_{3.0} \rightarrow c_{3.1}\ B_{3.2} \mid c_{4.1}$

$X_{5.0} \rightarrow D_{5.1}\ x_{5.2}$

$Y_{6.0} \rightarrow D_{6.1}\ y_{6.2}$

$D_{7.0} \rightarrow d_{7.1}\ D_{7.2} \mid d_{8.1}$

on voit en effet que :

$H_{k'}(XBccc) \cap H_{k'}(YBc) \ni d^{k'}$ quelque soit k' et que cette grammaire n'est pas COORD(n, q) quels que soient n et q puisque

$S[3.1] \cap S[4.1] \ni 3.1^{n}\ ccc>$

par contre cette grammaire est LR(1) car

$Z \overset{R*}{\rightarrow} <Yc^{p}ccc>$

$Z \overset{R*}{\rightarrow} <Xc^{p}ccc>$

avec

$<Yc^{p}ccc> = symb[0.1\ 3.1^{p+2}\ 4.1\ 0.3]$ et

$<Xc^{p}ccc> = symb[0.1\ 3.1^{p-1}\ 4.1\ 1.3\ 1.4\ 1.5\ 0.3]$

et on a évidemment

$<Yc^{p} \neq <Xc^{p}$

Résultat 4 Toute grammaire COORD(n, k) est LR(k)

Nous allons pour cela démontrer un résultat préalable; soit G une grammaire COORD(n, k); c'est-à-dire que

pour $\forall \overline{\varphi}, \overline{\varphi}', x, x'$ tels que

$Z \overset{R*}{\rightarrow} symb[\overline{\varphi}]\ x = \varphi x$ et

$Z \overset{R*}{\rightarrow} symb[\overline{\varphi'}]\ x' = \varphi' x'$

l'égalité $\overline{\varphi}$: n k : x = $\overline{\varphi}'$: n k : x' entraîne

ou bien a) $\overline{\varphi}$: 1 = c + e.d avec d < n(c + e) et

$\overline{\varphi}'$: 1 = f + h.g avec g < n(f + h)

et c + e.d ≡ f + h.g (mod. R_{symb})

ou bien b) $\overline{\varphi}$: 1 = $\overline{\varphi}'$: 1 = i + k.n(i + k)

(c + e.d, f + h.g, i + k.n(i + k) étant des éléments de C(G))

ce résultat préalable est alors le suivant :

si pour une CF grammaire G on a

$Z \overset{R*}{\rightarrow}$ symb$[\overline{\varphi}]$ x = Ψx

$Z \overset{R*}{\rightarrow}$ symb$[\overline{\varphi}']$ x' = Ψ' x'

avec m ≥ 1 le plus petit entier tel que

$\overline{\varphi}$: m ≠ $\overline{\varphi}'$: m

alors le plus petit entier p tel que

Ψ : p ≠ $\overline{\varphi}'$: p

est tel que p ≥ m et que p ne soit pas obligatoirement borné

Il est évident qu'on ne peut avoir p < m car on aurait alors

1 : (symb$[\overline{\varphi}]$: p) ≠ 1 : (symb$[\overline{\varphi}']$: p)

c'est-à-dire :

symb$[1 : (\overline{\varphi} : p)]$ ≠ symb$[1 : (\overline{\varphi}' : p)]$

or m est le plus petit entier tel que

$\overline{\varphi}$: m ≠ $\overline{\varphi}'$: m

donc p < m ⇒ $\overline{\varphi}$: p = $\overline{\varphi}'$: p

soit 1 : ($\overline{\varphi}$: p) = 1 : ($\overline{\varphi}'$: p) qui est contradictoire avec

symb$[1 : (\overline{\varphi} : p)]$ ≠ symb$[1 : (\overline{\varphi}' : p)]$

donc on a p ≥ m;

montrons qu'on peut avoir p > m : il suffit de considérer des grammaires où les règles sont du type suivant

$$A_{1.0} \rightarrow a_{1.1}\ B_{1.2} \mid C_{2.1}$$

$$B_{3.0} \rightarrow b_{3.1}\ B_{3.2} \mid b_{4.1}$$

$$C_{5.0} \rightarrow b_{5.1}\ C_{5.2} \mid b_{6.1}$$

on a

$$Z \overset{R*}{\rightarrow} < ab^p > = \text{symb}[0.1\ 1.1\ 3.1^{p-1}\ 4.1\ 0.3]$$

$$Z \rightarrow < b^p > = \text{symb}[0.1\ 5.1^{p-1}\ 6.1\ 0.3]$$

avec $\overline{\varphi}$ = 0.1 1.1 3.1^{p-1} 4.1 x = >

$\overline{\varphi}'$ = 0.1 5.1^{p-1} 6.1 x' = >

$symb[\overline{\varphi}] = < ab^p$

$symb[\overline{\varphi}'] = < b^p$

d'où m = 1 et p non borné

$A_{1.0} \rightarrow a_{1.1}\ b_{1.2}\ c_{1.3} \mid d_{2.1}\ b_{2.2}\ c_{2.3}$

on a

$Z \overset{R*}{\rightarrow} < a\ b\ c > = symb[0.1\ 1.1\ 1.2\ 1.3\ 0.3]$

$Z \overset{R*}{\rightarrow} < d\ b\ c > = symb[0.1\ 2.1\ 2.2\ 2.3\ 0.3]$

avec $\overline{\varphi}$ = 0.1 1.1 1.2 1.3 x = >

$\overline{\varphi}'$ = 0.1 2.1 2.2 2.3 x' = >

$symb[\overline{\varphi}] = < a\ b\ c$

$symb[\overline{\varphi}'] = < d\ b\ c$

d'où m = 1 et p = 3

On en déduit que pour toute grammaire COORD(n, k)

pour $\forall\, \overline{\varphi}, \overline{\varphi}'$, x, x' tels que

$Z \overset{R*}{\rightarrow} symb[\overline{\varphi}]\ x = \varphi x$

$Z \overset{R*}{\rightarrow} symb[\overline{\varphi}']\ x' = \varphi' x$

et $\overline{\varphi} : n\ k : x \neq \overline{\varphi}' : n\ k : x'$

on aura aussi pour $p \geq n$

$\varphi : p\ k : x \neq \varphi' : p\ k : x'$ et donc

$\varphi k : x \neq \varphi'\ k : x'$

cette dernière inégalité exprimant clairement que cette grammaire est LR(k)

Résultat 5 Toute grammaire BRC(n, k) est COORD(m, k) avec $m \leq n$

C'est une conséquence immédiate du résultat préalable au résultat 4 : en effet pour toute grammaire BRC(n, k),

pour $\forall \varphi, \varphi'$, x, x' tels que

$Z \overset{R*}{\rightarrow} symb[\overline{\varphi}]\ x = \varphi x$

$Z \overset{R*}{\rightarrow} symb[\overline{\varphi}']\ x' = \varphi'\ x'$

et $\varphi : n\ k : x \neq \varphi' : n\ k : x'$

on aura aussi pour $m \leq n$

$$\bar{\varphi} : m\ k : x \neq \bar{\varphi}' : m\ k : x'$$

cette dernière inégalité exprimant clairement que cette grammaire est COORD(m, k)

Résultat 6 Il existe des grammaires COORD(m, k) qui ne sont pas BRC(n, k) quelque soit n

appliquons enocre le résultat préalable au résultat 4 :
pour certaines grammaires COORD(m, k),

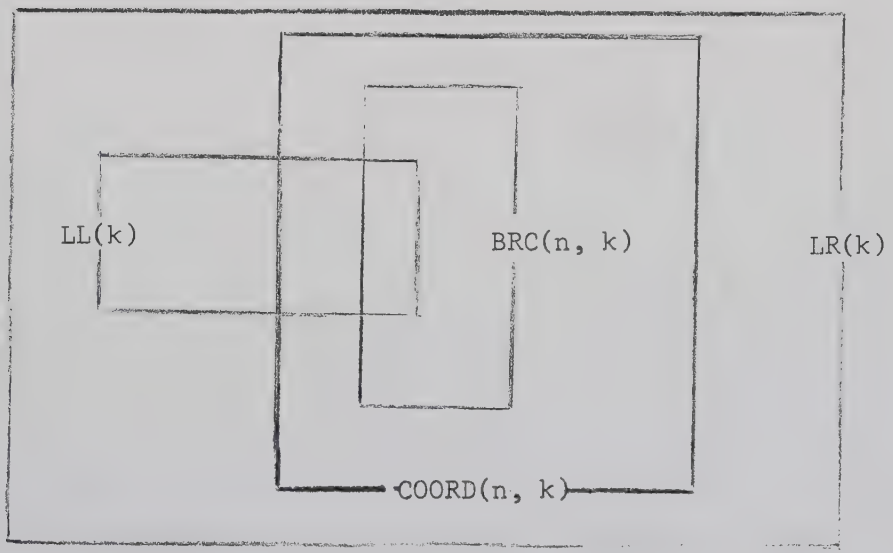
$\bar{\varphi}, \bar{\varphi}'$, x, x' tels que

$Z \overset{R*}{\rightarrow} \text{symb}[\bar{\varphi}]\ x = \varphi\ x$

$Z \overset{R*}{\rightarrow} \text{symb}[\bar{\varphi}']\ x' = \varphi'\ x$

et $\bar{\varphi} : m\ k : x \neq \bar{\varphi}' : m\ k : x'$
bien que $\varphi : n\ k : x = \varphi' : n\ k : x'$ quelque soit n borné; il s'agit du cas, dont un exemple a été donné, ou $n \geq m$ et n'est pas borné.

Tous ces résultats apparaissent sur le diagramme d'inclusion suivant :



On peut ajouter que les grammaires de précédence constituant l'ensemble BRC(1, 1), les grammaires COORD(1, 1) doivent pouvoir servir à la construction d'analyseurs de précédence moyennant un changement de l'alphabet de pile; l'intérêt serait d'obtenir la rapidité d'une méthode de précédence avec cependant un ensemble de grammaires admises plus grand.

CHAPITRE 7

UNE EXTENSION POSSIBLE DE LA METHODE COORD(m, k)

Considérons à nouveau la grammaire G_7 décrite dans l'exemple 11 page (34). On a :

$$R_7 = \{Z_{0.0} \rightarrow <_{0.1}\ S_{0.2}\ >_{0.3}$$
$$S_{1.0} \rightarrow x_{1.1}\ A_{1.2}\ a_{1.3} \mid y_{2.1}\ A_{2.2}$$
$$A_{3.0} \rightarrow a_{3.1}\ A_{3.2} \mid a_{4.1}\}$$

Cette grammaire, nous l'avons vue, n'est pas COORD(m, k) quels que soient m et k (et par contre est LL(3) et LR(3)).

Ceci est dû au fait qu'on a simultanément

$0.1\ 0.2\ 0.3 \overset{R*}{\rightarrow} 0.1\ 1.1\ (3.1)^n\ \boxed{4.1}\ 1.3\ 0.3$ et
$0.1\ 0.2\ 0.3 \rightarrow 0.1\ 2.1\ (3.1)^n\ \boxed{3.1}\ 4.1\ 0.3$

les symboles 3.1 et 4.1 dans un carré marquant que l'analyseur peut être dans les deux configurations suivantes

contexte gauche dans la pile	symbole en main en tête du ruban d'entrée E	contexte droit chaîne restante sur E
$0.1\ 1.1\ (3.1)^n$	a	a >
$0.1\ 2.1\ (3.1)^n$	a	a >

qui sont donc malheureusement identiques sur un contexte de longueur bornée à gauche et à droite alors que les décisions à prendre sont, pour le premier cas réduire 4.1 en 3.0 et dans le second cas empiler 3.1.

Cette situation donne l'idée de construire les contextes de façon différente que nous nous efforçons de décrire maintenant :

Soit une CF grammaire et un entier $k > 0$; considérons des chaînes de $C^*(G)$ $\overline{\varphi}_o$, $\overline{y}_o$ et $\overline{z}$ telles que :

α $\begin{cases} 0.1\ 0.2\ 0.3 \overset{R*}{\rightarrow} \overline{\varphi}_o\ \overline{y}_o\ \overline{z} \\ \text{avec symb}[\overline{\varphi}_o] \in V^*,\ \text{symb}[\overline{y}_o\ \overline{z}] \in T^*, \\ |\overline{y}_o| = q_o \geq k\ q_o \text{ tel que } |\text{symb}[\overline{y}_o]| = k \\ \text{On peut avoir deux situations} \end{cases}$

a) $\bar{\varphi}_o : 1 = i_o + k_o.j_o$ avec $j_o < n(i_o + k_o)$
et la situation correspondante pour l'analyseur est :
contenu de la pile P = $\bar{\varphi}_o$
symbole en main = 1 : symb$[\bar{y}_o]$
contexte droit = symb$[\bar{y}_o]$: (k - 1)

b) $\bar{\varphi}_o : 1 = i_o + k_o.n(i_o + k_o)$
et la situation correspondante pour l'analyseur est :
contenu de la pile P = $\bar{\varphi}_o$
symbole en main = symb$[\bar{\varphi}_o : 1]$
contexte droit = symb$[\bar{y}_o]$

D'autre part, on peut donner de $\bar{\varphi}_o$ l'expression suivante :

$$\bar{\varphi}_o = \underset{k=1}{\overset{k=r}{\text{CONC}}} c_k + e_k.1\ c_k + e_k.2 \ldots c_k + e_k.d_k$$

$\underset{k=1}{\overset{k=r}{\text{CONC}}}$ étant la concaténation ce que $\underset{k=1}{\overset{k=r}{\Sigma}}$ est à l'addition

avec $1 \leq d_k < n(c_k + e_k)$ pour $1 \leq k < r$
et $1 \leq d_r \leq n(c_r + e_r)$
ceci se déduit aisément de la relation $\overset{R}{\rightarrow}^*$; on a, avec ces notations

$$c_r + e_r.d_r = i_o + k_o.j_o = \bar{\varphi}_o : 1$$

Initialisons un ensemble CT i.e CT := si on est dans le cas a) alors $\emptyset$ sinon $\{i_o + k_o.n(i_o + k_o) - 1\}$

Considérons d'abord le cas a) envisagé plus haut
il existe une chaîne $\bar{y}'_o$ telle que :

$$\begin{cases} i_o + k_o.j_o + 1\ i_o + k_o.j_o + 2 \ldots i_o + k_o n(i_o + k_o) \overset{*}{\rightarrow} \bar{y}'_o \\ \text{avec } \bar{y}'_o \in C^*(G),\ \text{symb}[\bar{y}'_o] \in T^*,\ |\bar{y}'_o| \geq 1 \\ \text{et } \bar{y}'_o \text{ est un facteur gauche de } \bar{y}_o\ \bar{z} \end{cases}$$

effectuons CT := CT $\cup$ $\{i_o + k_o.j_o\}$
si nous avons $|\bar{y}'_o| \geq |\bar{y}_o|$ le calcul est définitivement terminé;
si par contre, nous avons $|\bar{y}'_o| < |\bar{y}_o|$,
nous posons $\bar{y}_o = \bar{y}'_o\ \bar{y}_1$: alors
il existe une chaîne $\bar{\varphi}_1$ telle que :

$0.1\ 0.2\ 0.3 \overset{R*}{\twoheadrightarrow} \bar{\varphi}_1\ \bar{y}_1\ \bar{z} \overset{R}{\twoheadrightarrow} \bar{\varphi}_o\ \bar{y}_o\ \bar{z}$

avec $\bar{\varphi}_1 \in C^*(G)$, $\text{symb}[\bar{\varphi}_1] \in V^*$,

$|\bar{\varphi}_1| \leq |\bar{\varphi}_o|$,

et $\bar{\varphi}_1$ et $\bar{\varphi}_o$ ont un facteur gauche en commun qui est

$$\underset{k=1}{\overset{k=r-1}{\text{CONC}}} \; c_k + e_k.1\ c_k + e_k.2 \ \ldots\ c_k + e_k.d_k = \bar{\varphi}_{o,\,1}$$

tel que

$\bar{\varphi}_o = \bar{\varphi}_{o,\,1}\ i_o + k_o.1\ i_o + k_o.2 \ \ldots\ i_o + k_o.j_o$ et

$\bar{\varphi}_1 = \bar{\varphi}_{o,\,1}\ c_r + e_r.d_r + 1$

et le calcul est terminé pour le cas a)

Considérons maintenant le cas b)
il existe une chaîne $\bar{\varphi}_1$ telle que

$0.1\ 0.2\ 0.3 \overset{R*}{\twoheadrightarrow} \bar{\varphi}_1\ \bar{y}_o\ \bar{z} \overset{R+}{\twoheadrightarrow} \bar{\varphi}_o\ \bar{y}_o\ \bar{z}$

avec $\bar{\varphi}_1 \in C^*(G)$, $\text{symb}[\bar{\varphi}_1] \in V^*$,

$\bar{\varphi}_1 : 1 = i_1 + k_1.j_1$ avec $j_1 < n(i_1 + k_1) - 1$

$|\bar{\varphi}_1| \leq |\bar{\varphi}_o|$

et $\bar{\varphi}_1$ et $\bar{\varphi}_o$ ont un facteur gauche en commun qui est

$$\underset{k=1}{\overset{k=r-\ell}{\text{CONC}}} \; c_k + e_k.1\ c_k + e_k.2 \ \ldots\ c_k + e_k.d_k = \bar{\varphi}_{o,\,1}$$

tel que :

$$\bar{\varphi}_o = \bar{\varphi}_{o,\,1} \underset{k=r-\ell+1}{\overset{k=r-1}{\text{CONC}}} \; c_k + e_k.1\ c_k + e_k.2 \ \ldots\ c_k + e_k.n(c_k + e_k) - 1$$

$i_o + k_o.1\ i_o + k_o.2 \ \ldots\ i_o + k_o.n(i_o + k_o)$ et

$\bar{\varphi}_1 = \bar{\varphi}_{o,\,1}\ c_{r-\ell} + e_{r-\ell}.d_{r-\ell} + 1$

et le calcul est terminé pour le cas b).

Considérons maintenant les issues des cas a) et b); à ce moment, on peut reformuler le même problème α) comme suit :
après traitement du cas a)

On considère des chaînes de $C^*(G)$ $\bar{\varphi}_1$, $\bar{y}_1$ et $\bar{z}$ telles que

$$\begin{cases} 0.1\ 0.2\ 0.3 \overset{R^*}{\Rightarrow} \bar{\varphi}_1\ \bar{y}_1\ \bar{z} \\ \text{avec } \mathrm{symb}[\bar{\varphi}_1] \in V^*,\ \mathrm{symb}[\bar{y}_1\ \bar{z}] \in T^*, \\ |\bar{y}_1| = q_1 < q_o \end{cases}$$

et après traitement du cas b)

$$\begin{cases} \text{On considère des chaînes de } C^*(G)\ \bar{\varphi}_1,\ \bar{y}_o \text{ et } \bar{z} \text{ telles que} \\ 0.1\ 0.2\ 0.3 \overset{R^*}{\Rightarrow} \bar{\varphi}_1\ \bar{y}_o\ \bar{z} \\ \text{avec } \mathrm{symb}[\bar{\varphi}_1] \in V^*,\ \mathrm{symb}[\bar{y}_o\ \bar{z}] \in T^*, \\ |\bar{y}_o| = q_o \end{cases}$$

Si donc nous revenons au problème initial, nous pouvons définir un algorithme donc chaque itération se compose des calculs du cas a) ou du cas b) si on reformule le problème à chaque nouvelle itération comme nous l'avons fait.

Montrons que cet algorithme s'achève toujours en un nombre fini d'itérations. Une itération du type cas a) succède soit à une itération de type cas a) soit à une itération de type cas b) et une itération de type cas b) ne peut succéder qu'à une itération de type cas a) (ou être la première itération); or chaque itération de type cas a) produit à partir d'une chaîne $\bar{y}_n$ une chaîne $\bar{y}_{n+1}$ telle $|\bar{y}_{n+1}| < |\bar{y}_n|$
puisque $|\bar{y}_o| = q_o$ on en déduit qu'il y au plus q_o itérations de type cas a) dans le déroulement de l'algorithme et d'après la règle de succession des itérations que nous avons donnée il y a aussi au plus q_o itérations de type cas b) dans le déroulement de l'algorithme.

Donnons maintenant la relation entre l'ensemble CT obtenu à la fin de l'algorithme et les chaînes $\bar{\varphi}_o$ et $\bar{y}_o$ du problème initial α).

Tout d'abord, on a

$$CT = \{f_o + h_o.g_o,\ f_1 + h_1.g_1,\ \ldots,\ f_1 + h_s.g_s\}$$

avec $0 \leq s < q_o$, $1 \leq g_o < n(f_o + h_o)$,
et $1 \leq g_t < n(f_t + h_t) - 1$ pour $1 \leq t \leq s$

Envisageons deux cas

1) l'algorithme a commencé sur une itération de type cas a) les itérations successives de type cas a) ont produit des chaînes $\overline{y}'_o, \overline{y}'_1, \ldots, \overline{y}'_s$ telles que

$$\begin{cases} \left(\underset{t=0}{\overset{t=s}{\text{CONC}}}\ \overline{y}'_t\right) z = \overline{y}_o\ \overline{z}, \quad \left|\underset{t=0}{\overset{t=s}{\text{CONC}}}\ \overline{y}'_t\right| \geqslant |\overline{y}_o| \\ \text{et } f_t + h_t.g_t + 1\ f_t + h_t.g_t + 2 \ldots f_t + h_t.n(f_t + h_t) \overset{*}{>} \overline{y}'_t \\ \text{pour } 0 \leqslant t \leqslant s \end{cases}$$

2) l'algorithme a commencé sur une itération de type cas b) les itérations successives de type cas a) ont produit des chaînes $\overline{y}'_o, \overline{y}'_1, \ldots, \overline{y}'_{s-1}$ telles que

$$\begin{cases} \left(\underset{t=0}{\overset{s-1}{\text{CONC}}}\ \overline{y}'_t\right) \overline{z} = \overline{y}_o\ \overline{z}, \quad \left|\underset{t=0}{\overset{t=s-1}{\text{CONC}}}\ \overline{y}'_t\right| \geqslant |\overline{y}_o| \\ \text{et } f_t + h_t.g_t + 1\ f_t + h_t.g_t + 2 \ldots f_t + h_t.n(f_t + h_t) \overset{*}{>} \overline{y}'_{t-1} \\ \text{pour } 1 \leqslant t \leqslant s \end{cases}$$

Dans les deux cas, les éléments de CT sont tels qu'on peut écrire

$$\begin{cases} \overline{\varphi}_o = \underset{t=s}{\overset{t=0}{\text{CONC}}}(\overline{\omega}_t\ f_t + h_t.g_t) \\ \text{avec } \overline{\omega}_t \in C^*(G) \text{ pour } 0 \leqslant t \leqslant s \\ \text{et le symbole } f_t + h_t.g_t \text{ ne fait pas partie de la chaîne } \overline{\omega}_{t-1} \text{ pour} \\ 1 \leqslant t \leqslant s \end{cases}$$

Après ces remarques préliminaires, nous pouvons dire, d'un point de vue intuitif, que le problème α) exprime toute configuration possible d'un analyseur ascendant en coordonnées, et que l'ensemble ordonné CT est l'ensemble des couples de coordonnées de la pile qui "interviennent" dans la formation de la chaîne terminale $\overline{y}_o$ de longueur k située en tête du ruban d'entrée.

Donc, soit une CF grammaire G et un entier $k > 0$, nous allons générer pour tout $i + k.j \in C(G)$ toutes les chaînes $\overline{z}'_t[i' + k.j]$ et tous les ensembles $CT_t[i + k.j]$ de la façon suivante :

soit toutes les chaînes $\overline{\varphi}$ et $\overline{z}$ telles que
0.1 0.2 0.3 $\overset{R*}{\Rightarrow} \overline{\varphi}\ i + k.j\ \overline{z}$ avec $symb[\overline{\varphi}\ i + k.j] \in V^*$
et $symb[\overline{z}] \in T^*$
alors $\overline{z}_t[i + k.j] = k' : \overline{z}$
avec k' tel que $|\ symb[[i + k.j]\ \overline{z}_t[i + k.j]]\ | = k + 1$
et $CT_t[i + k.j]$ est l'ensemble ordonné que calcule l'algorithme décrit précédemment pour le problème α) posé en considérant $\overline{z}_t[i + k.j]$ comme $\overline{y}_o$

Maintenant à chaque ensemble ordonné

$$CT_t[i + k.j] = \{f_o + h_o.g_o,\ f_1 + h_1.g_1,\ \ldots,\ f_s + h_s.g_s\}$$

nous faisons correspondre une chaîne

$$\overline{\ }_t[i + k.j] = \underset{i=s}{\overset{i=0}{CONC}}\ f_i + h_i.g_i$$

C'est à dessein que nous avons employé des notations identiques à celles du début du chapitre 3 car après avoir posé

$$\overline{\sigma}_t[i + k.j] = \overline{\varphi}_t[i + k.j]\ \ i + k.j\ \overline{z}_t[i + k.j]$$

pour $1 \leq t \leq \overline{ns}[i + k.j]$
nous proposons d'appliquer, avec cette nouvelle définition des chaînes $\overline{\sigma}_t[i + k.j]$ l'ensemble des opérations décrites dans le chapitre 4.

Nous allons conclure en décrivant un automate générateur pour ces nouvelles chaînes de contexte. Nous montrerons son fonctionnement pour la grammaire G_7 ainsi que l'analyseur qui peut en être déduit. Rappelons que nous reprenons les notations utilisées aux chapitres 4 et 5 (y compris les exemples) et que nous indiquerons les quelques points de différence.

L'automate générateur de chaînes sera $GE_2(i + k.j, G)$ (cf. chapitre 3). Ses états sont q_I, q_o, q_1, q_2, q_L, q_R, q_F et l'ensemble de ses instructions est :

$(q_J, \varepsilon) \rightarrow (q_o, \#i + k.j \#, \varepsilon, \varepsilon)$ [0]

$(q_o, \# i + k.n(i + k) \#) \rightarrow (q_o, \# i.o \#, \varepsilon, \varepsilon)$ [1]

$(q_o, \# i.o \#) \rightarrow (q_o, \# f + h.g \#, \varepsilon, \varepsilon)$ [2]

$(q_o, \# f + h.g \#) \rightarrow (q_1, \# f + h.g \#, \varepsilon, \varepsilon)$ [3]

$(q_1, \# f + h.g \#) \rightarrow (q_2, \# c + e.1\ c + e.2 \ldots c + e.d\ f + h.g \#\ c + e.d, \varepsilon)$ [4]

$(q_2, \# c + e.1\ c + e.2 \ldots c + e.n(c + e) \#) \rightarrow (q_L, \# c.o \#, \varepsilon, \varepsilon)$ [5]

$(q_2, \# c + e.1\ c + e.2 \ldots c + e.d\ f + h.g \#) \rightarrow (q_R, \# c + e.1\ c + e.2 \ldots c + e.d, f + h.g \#, \varepsilon, \varepsilon)$ [6]

$(q_L, \# c + e.1\ c + e.2 \ldots c + e.n(c + e) \#) \rightarrow (q_L, \# c.o \#, \varepsilon, \varepsilon)$ [7]

$(q_L, \# c.o \#) \rightarrow (q_L, \# f + h.g \#, \varepsilon, \varepsilon)$ [8]

$(q_L, \# f + h.g \#) \rightarrow (q_L, c + e.1\ c + e.2 \ldots c + e.d\ f + h.g \#, \varepsilon, \varepsilon)$ [9]

$(q_L, \# c + e.1\ c + e.2 \ldots c + e.d\ f + h.g \#) \rightarrow (q_R, \# c + e.1\ c + e.2 \ldots c + e.d\ f + h.g, c + e.d, \varepsilon)$ [10]

$(q_R, \# f + h.g\ c + e.1\ c + e.2 \ldots c + e.n(c + e) \#) \rightarrow (q_R, \# f + h.g\ c.o \#, \varepsilon, \varepsilon)$ [11]

$(q_R, \# f + h.g\ c.o \#) \rightarrow (q_R, \# f + h.g\ r + t.s \#, \varepsilon, \varepsilon)$ [12]

$(q_R, \# f + h.1\ f + g.2 \ldots f + h.n(f + h) \#) \rightarrow (q_L, \# f.o \#, \varepsilon, \varepsilon)$ [13]

$(q_R, \# f + h.g \#) \rightarrow (q_R, \# f + h.g\ c + e.d \#, \varepsilon, c + e.d)$ [14]

$(q_R, \varepsilon) \rightarrow (q_F, \varepsilon, \varepsilon, \varepsilon)$ [15]

$(q_L, \# o.o \#) \rightarrow (q_F, \varepsilon, \varepsilon, \varepsilon)$ [16]

avec les conditions suivantes

instruction [1] : $\forall$ i + k.n(i + k) tel que n(i + k) = 1

instruction [2] : $\forall$ f + h.g $\in$ i.o

instruction [3] : $\forall$ f + h.g tel qu'on n'ait pas g = n(f + h) = 1

instruction [4] : $\forall$ c + e.d tel que d < n(c + e) et f + h.g $\mathcal{R}$ c + e.d

instruction [8] : $\forall$ f + h.g $\in$ c.o tel que $\forall$ r + t.s $\in$ c.o on n'ait pas f + h.g $\mathcal{D}^{\dagger}$ r + t.s

instruction [9] : $\forall$ c + e.d tel que d < n(c + e) et f+ h.g $\mathcal{R}$ c + e.d

instruction [12] : $\forall$ r + t.s $\in$ c.o tel que r + t.s $\mathcal{R}$ f + h.g

instruction [14] : $\forall$ c + e.d tel que symb[c + e.d] $\in$ T et c + e.d $\mathcal{R}$ f + h.g

instruction [15] : dès que S_R contient q' symboles

instruction [16] : la génération ne peut plus continuer

Les tableaux ci-dessous donnent les suites de descriptions instantanées des générateurs correspondant à tous les $i + k.j \in C(G_7)$, ayant fait le choix $k = 3$ pour la longueur des chaînes terminales de "look-ahead";

$GE_2(3.1, 67)$

numéro de description	numéros des descriptions obtenues après application de l'instruction applicable	état	P	S_L	S_R	numéro de l'instruction applicable
1		q_I	ε	ε	ε	0
2		q_o	# 31 #	ε	ε	3
3	4, 18, 36	q_1	# 31 #	ε	ε	4
4		q_2	# 11 31 #	11	ε	6
5	6, 11	q_R	# 11 31 #	11	ε	14
6	7, 9	q_R	# 11 31 31 #	11	31	14
7		q_R	# 11 31 31 31 #	11	31 31	15
8		q_F	# 11 31 31 31 #	11	31 31	-
9		q_R	# 11 31 31 41 #	11	31 41	15
10		q_F	# 11 31 31 41 #	11	31 41	-
11		q_R	# 11 31 41 #	11	41	11
12		q_R	# 11 31 30 #	11	41	12
13		q_R	# 11 31 32 #	11	41	11
14		q_R	# 11 30 #	11	41	12
15		q_R	# 11 12 #	11	41	14
16		q_R	# 11 12 13 #	11	41 13	15
17		q_F	# 11 12 13 #	11	41 13	-
18		q_2	# 21 31 #	21	ε	6
19	20, 25	q_R	# 21 31 #	21	ε	14
20	21, 23	q_R	# 21 31 31 #	21	31	14
21		q_R	# 21 31 31 31 #	21	31 31	15
22		q_F	# 21 31 31 31 #	21	31 31	-
23		q_R	# 21 31 31 41 #	21	31 41	15
24		q_F	# 21 31 31 41 #	21	31 41	-
25		q_R	# 21 31 41 #	21	41	11
26		q_R	# 21 31 30 #	21	41	12
27		q_R	# 21 31 32 #	21	41	11
28		q_R	# 21 30 #	21	41	12
29		q_R	# 21 22 #	21	41	13
30		q_L	# 10 #	21	41	8
31		q_L	# 02 #	21	41	9
32		q_L	# 01 02 #	21	41	10
33		q_R	# 01 02 #	01 21	41	14
34		q_R	# 01 02 03 #	01 21	41 03	15
35		q_F	# 01 02 03 #	01 21	41 03	-
36		q_2	# 31 31 #	ε	ε	6
37	38, 43	q_R	# 31 31 #	31	ε	14
38	39, 41	q_R	# 31 31 31 #	31	31 31	14
39		q_R	# 31 31 31 31 #	31	31 31	15
40		q_F	# 31 31 31 31 #	31	31 41	-
41		q_R	# 31 31 31 41 #	31	31 41	15
42		q_F	# 31 31 31 41 #	31	31	-
43		q_R	# 31 31 41 #	31	41	11
44		q_R	# 31 31 30 #	31	41	12
45		q_R	# 31 31 32 #	31	41	11
46		q_R	# 31 30 #	31	41	12
47		q_R	# 31 32 #	31	41	13
48	49, 54	q_L	# 30 #	31	41	8

49		q_L	# 12 #	31	41	9
50		q_L	# 11 12 #	31	41	10
51		q_R	# 11 12 #	11 31	41	14
52		q_R	# 11 12 13 #	11 31	41 13	15
53		q_F	# 11 12 13 #	11 31	41 13	-
54		q_L	# 22 #	31	41	9
55		q_L	# 21 22 #	31	41	7
56		q_L	# 10 #	31	41	8
57		q_L	# 02 #	31	41	9
58		q_L	# 01 02 #	31	41	10
59		q_R	# 01 02 #	01 31	41	14
60		q_R	# 01 02 03 #	01 31	41 03	15
61		q_F	# 01 02 03 #	01 31	41 03	-

On a obtenu les chaînes suivantes

$\overline{\sigma}_1[3.1]$ = 1.1 $\boxed{3.1}$ 3.1 3.1 en ligne 8
$\overline{\sigma}_2[3.1]$ = 1.1 $\boxed{3.1}$ 3.1 4.1 en ligne 10
$\overline{\sigma}_3[3.1]$ = 1.1 $\boxed{3.1}$ 4.1 1.3 en ligne 17
$\overline{\sigma}_4[3.1]$ = 2.1 $\boxed{3.1}$ 3.1 3.1 en ligne 22
$\overline{\sigma}_5[3.1]$ = 2.1 $\boxed{3.1}$ 3.1 4.1 en ligne 24
$\overline{\sigma}_6[3.1]$ = 0.1 2.1 $\boxed{3.1}$ 4.1 0.3 en ligne 35
$\overline{\sigma}_7[3.1]$ = 3.1 $\boxed{3.1}$ 3.1 3.1 en ligne 40
$\overline{\sigma}_8[3.1]$ = 3.1 $\boxed{3.1}$ 3.1 4.1 en ligne 42
$\overline{\sigma}_9[3.1]$ = 1.1 3.1 $\boxed{3.1}$ 4.1 1.3 en ligne 53
$\overline{\sigma}_{10}[3.1]$ = 0.1 3.1 $\boxed{3.1}$ 4.1 0.3 en ligne 61

ce qui correspond à $CT_1[3.1] = \{1.1\}$ et $\overline{\varphi}_1[3.1] = 1.1$

$\vdots$

$CT_6[3.1] = \{2.1, 0.1\}$ et $\overline{\varphi}_6[3.1] = 0.1\ 2.1$

$\vdots$

$CT_{10}[3.1] = \{3.1, 0.1\}$ et $\overline{\varphi}_{10}[3.1] = 0.1\ 3.1$

numéro de description	numéros des descriptions obtenues après application de l'instruction applicable	état	P	S_L	S_R	numéro de l'instruction applicable
1		q_I	ε	ε	ε	0
2		q_o	# 41 #	ε	ε	1
3	4, 15, 24	q_o	# 30 #	ε	ε	2
4		q_o	# 12 #	ε	ε	3
5		q_1	# 12 #	ε	ε	4
6		q_2	# 11 12 #	11	ε	6
7		q_R	# 11 12 #	11	ε	14
8		q_R	# 11 12 13 #	11	13	13
9		q_L	# 10 #	11	13	8
10		q_L	# 02 #	11	13	9
11		q_L	# 01 02 #	11	13	10
12		q_R	# 01 02 #	01 11	13	14
13		q_R	# 01 02 03 #	01 11	13 03	15
14		q_F	# 01 02 03 #	01 11	13 03	-
15		q_o	# 22 #	ε	ε	3
16		q_1	# 22 #	ε	ε	4
17		q_2	# 21 22 #	21	ε	5
18		q_L	# 10 #	21	ε	8
19		q_L	# 02 #	21	ε	9
20		q_L	# 01 02 #	21	ε	10
21		q_R	# 01 02 #	01 21	ε	14
22		q_R	# 01 02 03 #	01 21	0.3	15
23		q_F	# 01 02 03 #	01 21	0.3	-
24		q_o	# 32 #	ε	ε	3
25		q_1	# 32 #	ε	ε	4
26		q_2	# 31 32 #	31	ε	5
27	28, 38	q_L	# 30 #	31	ε	8
28		q_L	# 12 #	31	ε	9
29		q_L	# 11 12 #	31	ε	10
30		q_R	# 11 12 #	11 31	ε	14
31		q_R	# 11 12 13 #	11 31	13	13
32		q_L	# 10 #	11 31	13	8
33		q_L	# 02 #	11 31	13	9
34		q_L	# 01 02 #	11 31	13	10
35		q_R	# 01 02 #	01 11 31	13	14
36		q_R	# 01 02 03 #	01 11 31	13 03	15
37		q_F	# 01 02 03 #	01 11 31	13 03	-
38		q_L	# 22 #	31	ε	9
39		q_L	# 21 22 #	31	ε	7
40		q_L	# 10 #	31	ε	8
41		q_L	# 02 #	31	ε	9
42		q_L	# 01 02 #	31	ε	10
43		q_R	# 01 02 #	01 31	ε	14
44		q_R	# 01 02 03 #	01 31	0.3	13
45		q_L	# 0.0 #	01 31	0.3	16
46		q_F	# 0.0 #	01 31	0.3	-

On a obtenu les chaînes suivantes :

$\overline{\sigma}_1[4.1]$ = 0.1 1.1 |4.1| 1.3 0.3 en ligne 14
$\overline{\sigma}_2[4.1]$ = 0.1 2.1 |4.1| 0.3 en ligne 23
$\overline{\sigma}_3[4.1]$ = 0.1 1.1 3.1 [4.1] 1.3 0.3 en ligne 37
$\overline{\sigma}_4[4.1]$ = 0.1 3.1 [4.1] 0.3 en ligne 46

numéro de description	numéros des descriptions obtenues après application de l'instruction applicable	état	P	S_L	S_R	numéro de l'instruction applicable
1		q_I	ε	ε	ε	0
2		q_o	# 21 #	ε	ε	3
3		q_1	# 21 #	ε	ε	4
4		q_2	# 01 21 #	ε	ε	6
5	6, 11	q_R	# 01 21 #	0.1	ε	14
6	7, 9	q_R	# 01 21 31 #	0.1	3.1	14
7		q_R	# 01 21 31 31 #	0.1	31 31	15
8		q_F	# 01 21 31 31 #	0.1	31 31	-
9		q_R	# 01 21 31 41 #	0.1	31 41	15
10		q_F	# 01 21 31 41 #	0.1	31 41	-
11		q_R	# 01 21 41 #	0.1	41	11
12		q_R	# 01 21 30 #	01	41	12
13		q_R	# 01 21 22 #	01	41	11
14		q_R	# 01 10 #	01	41	12
15		q_R	# 01 02 #	01	41	14
16		q_R	# 01 02 03 #	01	41 03	15
17		q_F	# 01 02 03 #	01	41 03	-

On a obtenu les chaînes suivantes $GE_2(2.1, G7)$

$\overline{\sigma}_1[2.1]$ = 0.1 [2.1] 3.1 3.1 en ligne 8
$\overline{\sigma}_2[2.1]$ = 0.1 [2.1] 3.1 4.1 en ligne 10
$\overline{\sigma}_3[2.1]$ = 0.1 [2.1] 4.1 1.3 en ligne 17

numéro de description	état	P	S_L	S_R	n° de l'instruction applicable
1	q_I	ε	ε	ε	0
2	q_o	# 1.3 #	ε	ε	3
3	q_1	# 1.3 #	ε	ε	4
4	q_2	# 1.1 1.2 1.3 #	1.2	ε	5
5	q_L	# 1.0 #	1.2	ε	8
6	q_L	# 0.2 #	1.2	ε	9
7	q_L	# 0.1 0.2 #	1.2	ε	10
8	q_R	# 0.1 0.2 #	0.1 1.2	ε	14
9	q_R	# 0.1 0.2 0.3 #	0.1 1.2	0.3	13
10	q_L	# 0.0 #	0.1 1.2	0.3	16
11	q_L	# 0.0 #	0.1 1.2	0.3	-

$GE_2(1.3, G7)$

On a obtenu la chaîne

$\overline{\sigma}_1(1.3)$ = 0.1 1.2 [1.3] 0.3 en ligne 11.

numéro de description	numéros des descriptions obtenues après application de l'instruction applicable	état	P	S_L	S_R	numéro de l'instruction applicable
1		q_I	ε	ε	ε	0
2		q_o	# 1.1 #	ε	ε	3
3		q_1	# 1.1 #	ε	ε	4
4		q_2	# 0.1 1.1 #	0.1	ε	6
5	6, 11	q_R	# 0.1 1.1 #	0.1	ε	14
6	7, 9	q_R	# 0.1 1.1 3.1 #	0.1	3.1	14
7		q_R	# 0.1 1.1 3.1 3.1 #	0.1	3.1 3.1	15
8		q_F	# 0.1 1.1 3.1 3.1 #	0.1	3.1 3.1	-
9		q_R	# 0.1 1.1 31 4.1 #	0.1	31 4.1	15
10		q_F	# 01 11 31 41 #	0.1	31 41	-
11		q_R	# 01 11 41 #	0.1	41	11
12		q_R	# 01 11 30 #	0.1	41	12
13		q_R	# 01 11 12 #	0.1	41	14
14		q_R	# 01 11 12 13 #	0.1	41 13	15
15		q_F	# 01 11 12 13 #	0.1	41 13	-

$GE_2(1.1, G7)$

On a obtenu les chaînes

$\overline{\sigma}_1[1.1]$ = 0.1 **1.1** 3.1 3.1 en ligne 8
$\overline{\sigma}_2[1.1]$ = 0.1 **1.1** 3.1 4.1 en ligne 10
$\overline{\sigma}_3[1.1]$ = 0.1 **1.1** 4.1 1.3 en ligne 15

numéro de description	état	P	S_L	S_R	numéro de l'instruction applicable
1	q_I	ε	ε	ε	0
2	q_o	# 0.2 #	ε	ε	3
3	q_1	# 0.2 #	ε	ε	4
4	q_2	# 0.1 0.2 #	0.1	ε	6
5	q_R	# 0.1 0.2 #	0.1	ε	14
6	q_R	# 0.1 0.2 0.3 #	0.1	0.3	13
7	q_L	# 0.0 #	0.1	0.3	16
8	q_F	# 0.0 #	0.1	0.3	-

$GE_2(0.2, G7)$

On a obtenu la chaîne

$\overline{\sigma}_1[0.2]$ = 0.1 **0.2** 0.3 en ligne 8

numéro de description	descriptions obtenues après application de l'instruction applicable	état	P	S_L	S_R	numéro de l'instruction applicable
1		q_I	ε	ε	ε	0
2		q_o	# 32 #	ε	ε	3
3		q_1	# 32 #	ε	ε	4
4		q_2	# 31 32 #	31	ε	5
5	6, 16	q_L	# 30 #	31	ε	8
6		q_L	# 12 #	31	ε	9
7		q_L	# 11 12 #	31	ε	10
8		q_R	# 11 12 #	11 31	ε	14
9		q_R	# 11 12 13 #	11 31	13	13
10		q_L	# 10 #	11 31	13	8
11		q_L	# 02 #	11 31	13	9
12		q_L	# 01 02 #	11 31	13	10
13		q_R	# 01 02 #	01 11 31	13	14
14		q_R	# 01 02 03 #	01 11 31	13 03	15
15		q_F	# 01 02 03 #	01 11 31	13 03	-
16		q_L	# 22 #	31	ε	9
17		q_L	# 21 22 #	31	ε	7
18		q_L	# 10 #	31	ε	8
19		q_L	# 02 #	31	ε	9
20		q_L	# 01 02 #	31	ε	10
21		q_R	# 01 02 #	01 31	ε	14
22		q_R	# 01 02 03 #	01 31	0.3	15
23		q_F	# 01 02 03 #	01 31	0.3	-

GE_2(3.2, G7)

On a obtenu les chaînes

$\overline{\sigma}_1[3.2]$ = 0.1 1.1 3.1 $\boxed{3.2}$ 1.3 0.3 en ligne 14

$\overline{\sigma}_2[3.2]$ = 0.1 3.1 $\boxed{3.2}$ 0.3 en ligne 23

numéro de description	état	P	S_L	S_R	numéro de l'instruction applicable
1	q_I	ε	ε	ε	0
2	q_o	# 22 #	ε	ε	3
3	q_1	# 22 #	ε	ε	4
4	q_2	# 21 22 #	21	ε	5
5	q_L	# 10 #	21	ε	8
6	q_L	# 02 #	21	ε	9
7	q_L	# 01 02 #	21	ε	10
8	q_R	# 01 02 #	01 21	ε	14
9	q_R	# 01 02 03 #	01 21	0.3	15
10	q_F	# 01 02 03 #	01 21	0.3	-

On a obtenu la chaîne GE_2(2.2, G7)

$\overline{\sigma}_1[2.2]$ = 0.1 2.1 $\boxed{2.2}$ 0.3 en ligne 10

numéro de description	état	P	S_L	S_R	numéro de l'instruction applicable
1	q_I	ε	ε	ε	0
2	q_o	# 12 #	ε	ε	3
3	q_1	# 12 #	ε	ε	4
4	q_2	# 11 12 #	11	ε	6
5	q_R	# 11 12 #	11	ε	14
6	q_R	# 11 12 13 #	11	13	13
7	q_L	# 10 #	11	13	8
8	q_L	# 02 #	11	13	9
9	q_L	# 01 02 #	11	13	10
10	q_R	# 01 02 #	01 11	13	14
11	q_R	# 01 02 03 #	01 11	13 03	15
12	q_F	# 01 02 03 #	01 11	13 03	-

On a obtenu la chaîne $GE_2(1.2, G7)$

$\overline{\sigma}_1[1.2]$ = 0.1 1.1 $\boxed{1.2}$ 1.3 0.3 en ligne 12.

Les générateurs $GE_2[0.1, G7]$ et $GE_2[0.3, G7]$ ont été omis car ils ne présentent pas d'intérêt.

Donc, pour la grammaire G_7, nous obtenons l'ensemble des chaînes $\overline{\sigma}_t[i + k.j]$ et $\hat{s}_t[i + k.j]$ dans le tableau ci-après.

	$\overline{\sigma}$ [i + k.j]			$\hat{s}_t$[i + k.j]		
t	$\overline{\varphi}_t$[i + k.j]	i + k.j	z_t[i + k.j]	$\overline{\rho}_t$[i + k.j]	cℓ[i + k.j]	r_t[i + k.j]
1	$\underline{\varepsilon}$	0.1	1.1 3.1	ε	<	xa
2	$\underline{\varepsilon}$	0.1	1.1 4.1	ε	<	xa
3	$\underline{\varepsilon}$	0.1	2.1 3.1	ε	<	ya
4	ε	0.1	$\underline{2}$.1 4.1	ε	<	ya
1	0.2	0.3	ε	0.2	>	ε
1	1.1	3.1	3.1 3.1	1.1	a	aa
2	1.1	3.1	3.1 4.1	1.1	a	aa
3	1.1	3.1	4.1 1.3	1.1	a	aa
4	2.1	3.1	3.1 3.1	2.1	a	aa
5	2.1	3.1	3.1 4.1	2.1	a	aa
6	0.1 2.1	3.1	4.1 0.3	0.1 2.1	a	a>
7	3.1	3.1	3.1 3.1	3.1	a	aa
8	3.1	3.1	3.1 4.1	3.1	a	aa
9	1.1 3.1	3.1	4.1 1.3	1.1 3.1	a	aa
10	0.1 3.1	3.1	4.1 0.3	0.1 3.1	a	a>
1	0.1 1.1	4.1	1.3 0.3	0.1 1.1	a	a>
2	0.1 2.1	4.1	0.3	0.1 2.1	a	>
3	0.1 1.1 3.1	4.1	1.3 0.3	0.1 1.1 3.1	a	a>
4	0.1 3.1	4.1	0.3	0.1 3.1	a	>
1	0.1	2.1	3.1 3.1	0.1	y	aa
2	0.1	2.1	3.1 4.1	0.1	y	aa
3	0.1	2.1	4.1 0.3	0.1	y	a>
1	0.1	1.1	3.1 3.1	0.1	x	aa
2	0.1	1.1	3.1 4.1	0.1	x	aa
3	0.1	1.1	4.1 1.3	0.1	x	aa
1	0.1 2.1	2.2	0.3	0.1 2.1	3.0	>
1	0.1 1.1	1.2	1.3 0.3	0.1 1.1	3.0	a>
1	0.1 1.1 3.1	3.2	1.3 0.3	0.1 1.1 3.1	3.0	a>
2	0.1 3.1	3.2	0.3	0.1 3.1	3.0	>
1	0.1	0.2	0.3	0.1	1.0	>
1	0.1 1.2	1.3	0.3	0.1 1.2	a	>

Nous voyons d'après ce tableau que la condition de déterminisme $\mathcal{C}_1$ du début du chapitre 4 est vérifiée. L'ensemble des chaînes $\hat{s}_t$[i + k.j] permet de construire une table de décision pour un analyseur déterministe des phrases de $L(G_7)$. Cet analyseur est un automate à deux piles $APD_2(G_7)$ = (Q, I, E, P1, P2).

P1 est une pile auxiliaire dont l'alphabet est {0.1, 1.1}; il a été construit à partir de symboles de $C(G_7)$ qui apparaissent dans les chaînes $\overline{\varphi}_t$[i + k.j] ailleurs qu'en dernière position à droite.

P2 est la pile principale dont l'alphabet est $C(G_7) \cup D(G)$.

I est un ensemble d'instructions représentées par des sextuplets (ℓc_1, ℓc_2, rc, co, rd, sh) avec

ℓc_1 est une chaîne sur l'alphabet de P1 de longueur $\leq k = 3$.

ℓc_2 est une chaîne de longueur 1 ou 2 sur l'alphabet de P1.

rc est une chaîne sur l'alphabet de E de longueur $\leq k = 3$.

co (coordonnées) est un élément $i + k.j \in C(G_7)$;

rd (réduire) et sh ("shift") sont deux variables booléennes.

L'analyseur interprète un sextuplet comme suit :

si (P_1) : $|\ell c_1| = \ell c_1$ et (P_2) : $|\ell c_2| = \ell c_2$ et
$|rc|$: (E) = rc alors
début
si $\ell c_2 : 1 \in D(G)$ alors effacer (P_2) : 1 de P_2;
écrire co sur P_2;
si co appartient à l'alphabet de P1 alors écrire co sur P1;
commentaire nous avons co = i + k.j;
si (P1) : 1 = i + k.j-1 alors effacer (P1) : 1 de P_1;
si rd alors
début
effacer (P_2) : j de P2;
commentaire co = i + k.j avec j = n(i +k);
si (P_1) : 1 = i + k.h avec $1 \leq h < n(i + k) - 1$ ou
(P2) : 1 = (P1) : 1 alors effacer (P1) : 1 de P1;
écrire i.0 sur P2;
fin;
si sh alors faire progresser E d'un symbole;
fin;

Nous pouvons maintenant donner la table de décision pour G_7.

numéro de ligne	ℓc1	ℓc2	rc	co	a_1	a_2
1			<xa	0.1		
2			<ya	0.1		
3		0.2	>	0.3		Réduire 0.0
4		1.1	aaa	3.1		
5		2.1	aaa	3.1		
6	0.1	2.1	aa>	3.1		
7		3.1	aaa	3.1		
8	1.1	3.1	aaa	3.1		
9	0.1	3.1	aa>	3.1		
10	0.1	1.1	aa>	4.1		Réduire 3.0
11	0.1	2.1	a>	4.1		Réduire 3.0
12	0.1 1.1	3.1	aa>	4.1		Réduire 3.0
13	0.1	3.1	a>	4.1		Réduire 3.0
14	0.1	1.2	a>	1.3		Réduire 1.0
15		0.1	yaa	2.1		
16		0.1	ya>	2.1		
17		0.1	xaa	1.1		
18	0.1	2.1 3.0	>	2.2	Ne pas avancer E	Réduire 1.0
19	0.1	1.1 3.0	a>	1.2	Ne pas avancer E	
20	0.1 1.1	3.1 3.0	a>	3.2	Ne pas avancer E	Réduire 3.0
21	0.1	3.1 3.0	>	3.2	Ne pas avancer E	Réduire 3.0
22		0.1 1.0	>	0.2	Ne pas avancer E	

Il est clair qu'on pourrait abréger cette table. Donnons la suite des configurations de l'analyseur pour les deux phrases de $L(G_7)$ <xaaaa> et <yaaaa>.

P1	P2	E	numéro de l'instruction applicable
$\overline{\varepsilon}$	$\overline{\varepsilon}$	<xaaa>	1
0.1	0.1	xaaaa>	17
0.1 1.1	0.1 1.1	aaaa>	4
0.1 1.1	0.1 1.1 3.1	aaa>	7
0.1 1.1	0.1 1.1 3.1 3.1	aa>	12
0.1 1.1	0.1 1.1 3.1 3.1 4.1	a>	
0.1 1.1	0.1 1.1 3.1 3.1 3.0	a>	20
0.1 1.1	0.1 1.1 3.1 3.1 3.2	a>	
0.1 1.1	0.1 1.1 3.1 3.0	a>	20
0.1 1.1	0.1 1.1 3.1 3.2	a>	
0.1	0.1 1.1 3.0	a>	19
0.1	0.1 1.1 1.2	a>	14
0.1	0.1 1.1 1.2 1.3	>	
$\overline{\varepsilon}$	0.1 1.0	>	22
$\overline{\varepsilon}$	0.1 0.2	>	3
$\overline{\varepsilon}$	0.1 0.2 0.3		
ε	0.0		
$\overline{\varepsilon}$	$\overline{\varepsilon}$	<yaaaa>	2
0.1	0.1	yaaaa>	15
0.1	0.1 2.1	aaaa>	5
0.1	0.1 2.1 3.1	aaa>	7
0.1	0.1 2.1 3.1 3.1	**aa**>	9
0.1	0.1 2.1 3.1 3.1 3.1	**a**>	13
0.1	0.1 2.1 3.1 3.1 3.1 4.1	>	
0.1	0.1 2.1 3.1 3.1 3.1 3.0	>	21
0.1	0.1 2.1 3.1 3.1 3.1 3.2	>	
0.1	0.1 2.1 3.1 3.1 3.0	>	21
0.1	0.1 2.1 3.1 3.1 3.2	>	
0.1	0.1 0.1 2.1 3.1 3.0	>	2.1
0.1	0.1 2.1 3.1 3.2	>	
0.1	0.1 2.1 3.0	>	18
0.1	0.1 2.1 2.2	>	
$\overline{\varepsilon}$	0.1 1.0	>	22
$\overline{\varepsilon}$	0.1 0.2	>	3
$\overline{\varepsilon}$	0.1 0.2 0.3		
ε	0.0		

Nous nous proposons d'étudier la méthode que nous venons de décrire sur cet exemple. Nous pensons que l'ensemble des CF grammaires qu'elle accepte contient strictement les ensembles LL(k) et COORD(m, k). Une question ouverte est : cette méthode accepte-t-elle l'ensemble des grammaires LR(k) ? si la réponse est positive, on disposerait alors d'une méthode telle que la complexité de la table de décision dépende de la grammaire proposée : par exemple, si la grammaire proposée est COORD(1, 1), cette méthode pourrait alors donner une table de décision réduite (cf. exemple 13) où il n'est jamais nécessaire de consulter la pile auxiliaire P1.

Rappelons le résultat essentiel : pour un automate d'analyse ascendante dont l'alphabet de pile est l'ensemble C(G), toute chaîne de k symboles terminaux située en tête du ruban d'entrée a pour "contexte" à gauche k symboles au plus dans la pile (si G n'a pas de productions vides). La pile auxiliaire que nous avons définie contient donc le résumé utile de la pile principale et permet d'éviter ce qui fait échouer sur certaines grammaires, les méthodes BRC(m, k) et COORD(m, k), à savoir des contextes à gauche "inaccessibles". Cette méthode fera l'objet de nos travaux ultérieurs car bien des aspects n'en sont pas encore clairs.

QUELQUES EXEMPLES PROGRAMMATION (*)

Les algorithmes décrits pour l'ensemble des CF grammaires COORD(m, k) ont été programmés en PL/1 sur IBM 360. Le choix de ce langage est dû aux facilités qu'il offre pour la mise en oeuvre d'algorithmes dits "non numériques". Nous n'avons cependant pas utilisé toutes les possibilités de PL/1 : en particulier, au prix d'une certaine perte éventuelle de temps, la gestion dynamique de la mémoire s'effectue systématiquement en utilisant la structure de bloc; les zones variables dont il est impossible de calculer la longueur sont des tableaux à bornes variables déclarés en tête de bloc; si au cours de calcul, on constate un débordement, on sort du bloc, on augmente les bornes du tableau concerné d'un certain incrément et on rentre de nouveau dans le bloc en recommençant le calcul interrompu; ceci nécessite que chaque zone variable soit accompagnée d'un système de pointeurs à gérer en permanence. Le résultat est que le programme n'est pas extrêmement performant au point de vue temps mais présente une bonne "résistance" aux utilisateurs qui ne font pas toujours des prévisions exactes quant à l'espace nécessaire pour leurs calculs ... .

L'entrée des données a été simplifiée autant que possible et se réduit pour les grammaires d'une taille raisonnable à l'ensemble des productions dans la convention d'écriture BNF; pour les autres cas un système de paramètres facultatifs à valeurs par défaut a été utilisé en s'inspirant du JCL d'IBM. De cette façon l'utilisation effective du programme ne nécessite pas un long apprentissage.

Un des paramètres facultatifs permet d'obtenir tous les résultats intermédiaires mais généralement seule est imprimée une table de décision correspondant à la grammaire entrée. Si cette grammaire n'est pas COORD(m, k), un diagnostic est fourni et les chaînes de contexte incriminées sont imprimées ce qui permet de réécrire les règles dans la forme convenable. Trois sortes de tables de décision peuvent être imprimées :

a) table complète de tous les contextes (identifiée par le titre "lengthy table") à consultation aléatoire; elle est en général fort encombrante et ne présente d'intérêt que pour une détection immédiate des erreurs syntaxiques.

(*) Je remercie Madame CONNES et Monsieur SALZEDO de m'avoir donné accès au 360/75 du C.I.R.C.E.

b) table courte des contextes (identifiée par le titre "short random table") à consultation aléatoire; les contextes y sont réduits à la longueur minimale pour conserver la propriété de déterminisme et un ordre de consultation quelconque; cette table est substantiellement plus courte que la précédente mais la détection des erreurs syntaxiques est bien sûr parfois retardée.

c) table courte des contextes (identifiée par le titre "short sequential table" à consultation séquentielle; les contextes y sont réduits à la longueur minimale pour conserver la propriété de déterminisme à la condition que, pour un symbole en main donné, la consultation se fasse dans l'ordre d'écriture de la table. L'encombrement est minimal par contre le temps de consultation est plus élevé et la détection des erreurs est éventuellement retardée.

C'est à l'utilisateur de choisir la solution qui lui semble la mieux adaptée à ses besoins.

Chaque ligne d'une table de décision apparait sous la forme suivante : contenu de pile, barre verticale, contenu du ruban d'entrée, entre parenthèses le symbole à écrire sur la pile, éventuellement l'indication "no shift" montrant que le ruban d'entrée ne doit pas avancer d'un symbole, éventuellement l'indication "reduce" suivie d'un symbole i.0 montrant qu'une production complète i + k.1 i + k.2 ... i + k.n(i + k) se trouve en tête de la pile et doit être réduite en le symbole i.0 et enfin entre parenthèses l'indice de la chaîne de l'ensemble CX(G, m, k) ayant servi à construire la ligne de la table de décision.

Avant la table de décision, les règles de la grammaire sont listées sous une forme très proche de la notation BNF à ceci près que le symbole → est remplacé par le symbole : et que chaque symbole de V apparaît précédé par l'élément correspondant de $C(G) \cup D(G)$ entre parenthèses; de plus la chaîne vide est représentée par le symbole nostring.

Une liste des productions à réduire termine les résultats; elle permet de vérifier la correction des productions à réduire dans le cas où on utilise une table "courte" des contextes.

B I B L I O G R A P H I E

{ALP 71} R. ALPIAR; Double syntax oriented processing, The Computer Journal Vol 14, number 1, 1971.

{BOR 71} J. BORDIER; Méthodes pour la mise au point de grammaires LL(1); Thèse 3ème cycle; Grenoble 1971.

{COL 67} A. COLMERAUER; Total precedence relations; J.A.C.M.; Janvier 1970.

{COU 68} J. COURTIN; Langages analysables de gauche à droite, construction d'un analyseur pour langages LR(1); Thèse 3ème cycle; Grenoble 1968.

{EIC 64} J. EICKEL; Génération of parsing algorithms for Chomsky 2-type languages; Technische Hochschule Bericht nr. 6401; Münich 1964.

{FEL 68} J. FELMAN, D. GRIES; Translator Writing Systems; C.A.C.M.; Février 1968.

{FLO 64} R.W. FLOYD; Bounded context syntactic analysis; C.A.C.M.; Février 1964.

{FLO 63} R.W. FLOYD; Syntactic analysis and operator precedence; J.A.C.M.; Juillet 1963.

{GRI 69} M. GRIFFITHS; Analyse déterministe et compilateurs; Thèse d'état; Grenoble 1969.

{KNU 67} D.E. KNUTH; Top down syntax analysis; International summer school on computer programming; Copenhague 1967.

{KNU 65} D.E. KNUTH; On the translation of languages from left to right; Information and Control; Décembre 65.

{KOR 69} A.J.A. KORENJAK; A practical method for constructing LR(k) processors; C.A.C.M.; Novembre 1969.

{LOE 70} J.J. LOECKX; An algorithm for the construction of bounded context parsers; C.A.C.M.; Mai 1970.

{LOE 68} J.J. LOECKX; Mecanical construction of bounded context parsers for Chomsky 0-type languages; Thèse d'état; Louvain 1968.

{TER 71} G. TERRINE; An algorithm generating the decision table of a deterministic bottom-up parser for a subset of context free grammars; Proceedings of the third annual symposium on theory of computing; A.C.M. Mai 1971.

{VDB 70} J. VAN DEN BROEK; An algorithm for the bounded context parser or for a coordinate parser; rapport de stage; université de technologie d'Enschede (Pays Bas); Juillet 1970.

ANNEXES

REPERTOIRE DES PRINCIPALES NOTATIONS ET TERMINOLOGIES

page de première occurrence

7 N : ensemble des symboles auxiliaires d'une CF grammaire G

7 T : ensemble des symboles terminaux d'une CF grammaire G

7 R : ensemble des règles d'une CF grammaire G

7 Z : symbole initial ou axiome d'une CF grammaire G

7 A^* : ensembles des chaînes de longueur finie sur A ou monoïde sur A

7 $\rightarrow$: relation binaire $\subset V^* \times V^*$ définie par R l'ensemble des règles
8 de G, exprimant toutes les dérivations licites pour produire les phrases de G et le langage L(G) à partir de Z

8 $\overset{+}{\rightarrow}$: relation binaire transitive $\subset V^* \times V^*$ construite par fermeture transitive de la relation $\rightarrow$, exprimant toutes les séquences de dérivations licites pour produire les phrases de G i.e. PH(G) y compris le langage de G i.e. L(G)

8 $\overset{*}{\rightarrow}$: relation binaire transitive et réflexive $\subset V^* \times V^*$ construite par fermeture transitive et réflexive de la relation $\rightarrow$, exprimant toutes les séquences de dérivations licites (y compris la séquence vide) pour produire PH(G)

8 $\overset{L}{\rightarrow}$, $\overset{L+}{\rightarrow}$, $\overset{L*}{\rightarrow}$, $\overset{R}{\rightarrow}$, $\overset{R+}{\rightarrow}$, $\overset{R*}{\rightarrow}$: relations binaires $\subset V^* \times V^*$ exprimant les dérivations et séquences de dérivations par réécriture du symbole auxiliaire le plus à gauche (représentation par L) ou le plus à droite (représentation par R) afin de produire les ensembles PHD(G) y compris L(G) ou PHA(G) y compris L(G)

8 PH(G) : ensemble des phrases de G y compris L(G) engendré à l'aide des relations $\rightarrow$, $\overset{+}{\rightarrow}$, $\overset{*}{\rightarrow}$

8 PHD(G) : ensemble des phrases descendantes de G, y compris L(G), engendré à l'aide des relations $\overset{L}{\rightarrow}$, $\overset{L+}{\rightarrow}$, $\overset{L*}{\rightarrow}$

8 PHA(G) : ensemble des phrases ascendantes de G, y compris L(G), engendré à l'aide des relations $\overset{R}{\rightarrow}$, $\overset{R+}{\rightarrow}$, $\overset{R*}{\rightarrow}$

8 L(G) : ensemble des phrases sur l'alphabet T de G

8 ASD(x, G) : ensemble ordonné de description descendante d'un arbre syntaxique d'une phrase x de L(G)

8 ASA(x, G) : ensemble ordonné de description ascendante d'un arbre syntaxique d'une phrase x de L(G)

10 $k:\varphi$: application de $N \times V^* \to \bigcup_{i \leq k} V^i$ dont la valeur pour un entier k et une chaîne φ est une chaîne ψ facteur gauche de φ telle que $\varphi = \psi\theta$ et $|\psi| = \mathrm{Min}(k, |\varphi|)$

10 $\varphi : k$: application de $V^* \times N \to \bigcup_{i \leq k} V^i$ dont la valeur pour une chaîne φ et un entier k est une chaîne ψ facteur droit de φ telle que $\varphi = \theta\psi$ et $|\psi| = \mathrm{Min}(k, |\varphi|)$

11 règle initiale : règle ayant une production $\vdash S \dashv$ dont le symbole initial ou axiome Z est le sujet

13 C(G) : alphabet de coordonnées défini à partir des productions G

13 GC : grammaire G réécrite sur l'alphabet C(G) en place de V

13 i + k.j : élément de C(G) associé à un symbole de V apparaissant en $j^{\text{ème}}$ position de la $(k+1)^{\text{ème}}$ production de la règle dont le sujet est A lui-même associé au symbole i.0 de D(G)

14 D(G) : alphabet de coordonnées distinct de C(G), défini à partir des sujets des règles de G

14 ℓ.0 : élément de D(G) associé au sujet d'une règle de G, tel que la $(\ell-1)^{\text{ème}}$ production soit la dernière production de la règle précédente étant entendu que le sujet Z est associé à 0.0

14 i.j : autre façon d'écrire un symbole de G lorsqu'on ne veut pas expliciter le numéro de la production par rapport au sujet de la règle

15 W : $W = V - \{Z\}$

15 symb : homomorphisme $C^*(G) \to W^*$

15 E(G) : $E(G) = D(G) - \{0.0\}$

16 $\overset{}{\twoheadrightarrow}$, $\overset{+}{\twoheadrightarrow}$, $\overset{*}{\twoheadrightarrow}$, $\overset{L}{\twoheadrightarrow}$, $\overset{L+}{\twoheadrightarrow}$ etc : relations binaires homologues de $\to$, $\overset{+}{\to}$, $\overset{*}{\to}$, $\overset{L}{\to}$, $\overset{L+}{\to}$ mais par rapport à la grammaire GC et définissant des sous-ensembles de $C(G) \times C(G)$

21 AP(G) : automate analyseur syntaxique ascendant et non forcément déterministe pour le langage L(G)

21 $\hat{\text{ASA}}(x, G)$: ensemble ordonné de description ascendante, exprimé par des chaînes dont le facteur gauche réduit est exprimé en termes de C(G), d'un arbre syntaxique d'une phrase x de L(G); cet ensemble fournit la même information que ASA(x, G)

24 $\overline{\sigma}_t[i + k.j] = \overline{\varphi}_t[i + k.j]\ i + k.j\ \overline{z}_t[i + k.j]$: forme générale de la $t^{\text{ème}}$ chaîne de contexte borné à m symboles à gauche et q' à droite, appartenant à $\bigcup_{i \leq m+q'+1} C^i(G)$, pour le symbole i + k.j et exprimant une configuration possible d'un analyseur ascendant AP(GC)

24 $\hat{s}_t[i + k.j] = \overline{\varphi}_t[i + k.j] c\ell[i + k.j]\ r_t[i + k.j]$: forme générale de la $t^{\text{ème}}$ chaîne de contexte borné à m symboles à gauche et q à droite, appartenant à $(\bigcup_{i \leq m} C^i(G))\ (T \cup E(G))\ (\bigcup_{i \leq k} T^i)$ et exprimant une configuration possible d'un analyseur ascendant AP(G)

26 $S[i + k.j]$: ensemble $\overline{\sigma}_t[i + k.j]$

26 $\mathcal{N}$: relation binaire $\subset C(G) \times C(G)$ exprimant une succession possible entre deux symboles de C(G) dans les phrases de PH(Gc)

26 $\mathcal{P}$: relation binaire $\subset C(G) \times C(G)$, exprimant le fait qu'un symbole de C(G) soit le symbole de tête d'une phrase dérivée du deuxième symbole

26 $\mathcal{D}$: relation binaire $\subset C(G) \times C(G)$, exprimant le fait qu'un symbole de C(G) soit le symbole de tête d'une phrase dérivée du deuxième symbole

28 GE(i + k.j, G) : automate générant l'ensemble $S[i + k.j]$ pour G

32 U(f.O) : ensemble des productions de sujet f.O dans Gc

32 CX(G, m, k) : ensemble des ensembles $S[i + k.j]$ et U(f.O) pour tous les i + k.j de C(G) et tous les f.O de D(G)

34 $\mathcal{C}_1$: condition de déterminisme valable pour un sous-ensemble particulier des grammaires COORD(m, q) défini par J.J. LOECKX

35 $\frac{x}{A}$: chaîne x écrite sur l'alphabet A

35 $\frac{E}{A}$: ensemble E de chaînes écrites sur l'alphabet A

35 $\mathcal{R}_1$: relation binaire $\subset C(G) \times C(G)$ exprimant que deux éléments de C(G) peuvent être codés de la même façon dans la pile de AP(G) pour produire ainsi APD(G)

36 $C^{(a)}(G)$: ensemble $C^{(a-1)}/\mathcal{R}_1^+$ avec $C^{(o)}(G) \triangleq C(G)$

36 $CX^{(a)}(G, m, q)$: ensemble $CX^{(a-1)}(G, m, q)$ ré-écrit sur l'alphabet $C^{(a)}(G)$ avec $CX^{(o)}(G, m, q) \triangleq CX(G, m, q)$

36 $\mathcal{C}$: condition de déterminisme valable pour l'ensemble des grammaires COORD(m, k)

45 APD(G) : automate analyseur syntaxique ascendant et déterministe associé à une grammaire COORD(m, q)

67 $\underset{i=1}{\overset{n}{\text{CONC}}}\, y_i \triangleq y_1\, y_2 \cdots y_n$

67 $CT_t[i + k.j]$: ensemble ordonné d'éléments de C(G) constituant le résumé "utile" de la pile de l'analyseur quand le symbole en main est cℓ $[i + k.j]$

71 $GE_2(i + k.j, G)$: automate générant les chaînes de contexte $\overline{\sigma}_t[i + k.j]$ pour la grammaire G; à la différence de GE(i + k.j, G) le facteur gauche $\overline{\varphi}_t[i + k.j]$ de ces chaînes représente le résumé "utile" de la pile de l'analyseur formé à partir de $CT_t[i + k.j]$

```
GO                          TABLE COMPLETE A ACCES ALEATOIRE(<=1,<=1)
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1


                                                          BEGINGRAM
Z   .<. R .>.                                                    *
R   S M ... M                                                    *
S   .+. | .-. | .NOSTRING.                                      *
M   N | .NOSTRING.                                              *
M   N .D. | .D.
                                                           ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z       : (  0.1) <       (  0.2) R       (  0.3) >

(  1.0) R       : (  1.1) S       (  1.2) M       (  1.3) .       (  1.4) M

(  2.0) S       : (  2.1) +
(  2.0) S       : (  3.1) -
(  2.0) S       : (  4.1) NOSTRING

(  5.0) M       : (  5.1) N
(  5.0) M       : (  6.1) NOSTRING

(  7.0) N       : (  7.1) N       (  7.2) D
(  7.0) N       : (  8.1) D
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.0 WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.0:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

LENGTHY TABLE

	|<	+	(0.1)				(1)
	|<	-	(0.1)				(2)
	|<	.	(0.1)				(3)
	|<	D	(0.1)				(4)
0.2	|>		(0.3)		REDUCE	0.0	(5)
1.2	|.	>	(1.3)				(6)
1.2	|.	D	(1.3)				(7)
0.1	|+	.	(2.1)		REDUCE	2.0	(8)
0.1	|+	D	(2.1)		REDUCE	2.0	(9)
0.1	|-	.	(3.1)		REDUCE	2.0	(10)
0.1	|-	D	(3.1)		REDUCE	2.0	(11)
0.1	|.	>	(4.1)	NO SHIFT	REDUCE	2.0	(12)
0.1	|D	.	(4.1)	NO SHIFT	REDUCE	2.0	(13)
0.1	|D	D	(4.1)	NO SHIFT	REDUCE	2.0	(14)
0.1	|.	D	(4.1)	NO SHIFT	REDUCE	2.0	(15)
1.1	|.	>	(6.1)	NO SHIFT	REDUCE	5.0	(16)
1.3	|>		(6.1)	NO SHIFT	REDUCE	5.0	(17)
1.1	|.	D	(6.1)	NO SHIFT	REDUCE	5.0	(18)
7.1	|D	.	(7.2)		REDUCE	7.0	(19)
7.1	|D	D	(7.2)		REDUCE	7.0	(20)
7.1	|D	>	(7.2)		REDUCE	7.0	(21)
1.1	|D	.	(8.1)		REDUCE	7.0	(22)
1.1	|D	D	(8.1)		REDUCE	7.0	(23)
1.3	|D	>	(8.1)		REDUCE	7.0	(24)
1.3	|D	D	(8.1)		REDUCE	7.0	(25)
1.1	7.0 | .		(5.1)	NO SHIFT	REDUCE	5.0	(26)
1.3	7.0 | >		(5.1)	NO SHIFT	REDUCE	5.0	(27)
1.1	7.0 | D		(7.1)	NO SHIFT			(28)
1.3	7.0 | D		(7.1)	NO SHIFT			(29)
1.1	5.0 | .		(1.2)	NO SHIFT			(30)
1.3	5.0 | >		(1.4)	NO SHIFT	REDUCE	1.0	(31)
0.1	2.0 | .		(1.1)	NO SHIFT			(32)
0.1	2.0 | D		(1.1)	NO SHIFT			(33)
0.1	1.0 | >		(0.2)	NO SHIFT			(34)

*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3	
1.0	1.1	1.2	1.3	1.4
2.0	2.1			
2.0	3.1			
2.0	4.1			
5.0	5.1			
5.0	6.1			
7.0	7.1	7.2		
7.0	8.1			

```
GO                                        TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1



                                                                  BEGINGRAM
Z   .<. R .>.                                                         *
R   S M ... M                                                         *
S   .+. | .-. | .NOSTRING.                                           *
M   N | .NOSTRING.                                                   *
N   N .D. | .D.
                                                                  ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z      : (  0.1) <      (  0.2) R      (  0.3) >

(  1.0) R      : (  1.1) S      (  1.2) M      (  1.3) .      (  1.4) M

(  2.0) S      : (  2.1) +
(  2.0) S      : (  3.1) -
(  2.0) S      : (  4.1) NOSTRING

(  5.0) M      : (  5.1) N
(  5.0) M      : (  6.1) NOSTRING

(  7.0) N      : (  7.1) N      (  7.2) D
(  7.0) N      : (  8.1) D
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<		(0.1)				(1)
0.2	|>		(0.3)		REDUCE	0.0	(5)
1.2	|.		(1.3)				(6)
	|+		(2.1)		REDUCE	2.0	(8)
	|-		(3.1)		REDUCE	2.0	(10)
0.1	|.		(4.1)	NO SHIFT	REDUCE	2.0	(12)
0.1	|>		(4.1)	NO SHIFT	REDUCE	2.0	(13)
1.1	|.		(6.1)	NO SHIFT	REDUCE	5.0	(16)
1.3	|>		(6.1)	NO SHIFT	REDUCE	5.0	(17)
7.1	|D		(7.2)		REDUCE	7.0	(19)
1.1	|D		(8.1)		REDUCE	7.0	(22)
1.3	|D		(8.1)		REDUCE	7.0	(24)
	7.0	| .	(5.1)	NO SHIFT	REDUCE	5.0	(26)
	7.0	| >	(5.1)	NO SHIFT	REDUCE	5.0	(27)
	7.0	| D	(7.1)	NO SHIFT			(28)
1.1	5.0	|	(1.2)	NO SHIFT			(30)
1.3	5.0	|	(1.4)	NO SHIFT	REDUCE	1.0	(31)
	2.0	|	(1.1)	NO SHIFT			(32)
	1.0	|	(0.2)	NO SHIFT			(34)

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3	
1.0	1.1	1.2	1.3	1.4
2.0	2.1			
2.0	3.1			
2.0	4.1			
5.0	5.1			
5.0	6.1			
7.0	7.1	7.2		
7.0	8.1			

G1 (KNU65) P. 613 TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)

```
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1
```

```
                                                        BEGINGRAM
Z   .<. S .>.                                               *
S   .A. A .C. | .B.                                         *
A   .A. S .C. | .B.
                                                        ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z          : (  0.1) <          (  0.2) S          (  0.3) >

(  1.0) S          : (  1.1) A          (  1.2) A          (  1.3) C
(  1.0) S          : (  2.1) B

(  3.0) A          : (  3.1) A          (  3.2) S          (  3.3) C
(  3.0) A          : (  4.1) B
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<	(0.1)				(1)
	|>	(0.3)		REDUCE	0.0	(3)
0.1	|A	(1.1)				(4)
3.1	|A	(1.1)				(5)
1.1	|A	(3.1)				(8)
1.2	|C	(1.3)		REDUCE	1.0	(10)
3.2	|C	(3.3)		REDUCE	3.0	(12)
0.1	|B	(2.1)		REDUCE	1.0	(13)
3.1	|B	(2.1)		REDUCE	1.0	(14)
1.1	|B	(4.1)		REDUCE	3.0	(15)
3.0	1.0 |	(1.2)	NO SHIFT			(16)
0.1	1.0 |	(0.2)	NO SHIFT			(17)
3.1	1.0 |	(3.2)	NO SHIFT			(18)

*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3
1.0	1.1	1.2	1.3
1.0	2.1		
3.0	3.1	3.2	3.3
3.0	4.1		

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z          : (  0.1) <          (  0.2) S          (  0.3) >

(  1.0) S          : (  1.1) A          (  1.2) A
(  1.0) S          : (  2.1) B          (  2.2) B

(  3.0) A          : (  3.1) C          (  3.2) A
(  3.0) A          : (  4.1) D

(  5.0) B          : (  5.1) C          (  5.2) B
(  5.0) B          : (  6.1) D
```

G3 (FEL68) P. 79 TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)

```
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1
```

```
Z  .<. S .>.
S  S .+. T | T
T  T .*. F | F
F  .(. S .). | .V.
```

```
BEGINGRAM
*
*
*

ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z          : (  0.1) <          (  0.2) S          (  0.3) >

(  1.0) S          : (  1.1) S          (  1.2) +          (  1.3) T
(  1.0) S          : (  2.1) T

(  3.0) T          : (  3.1) T          (  3.2) *          (  3.3) F
(  3.0) T          : (  4.1) F

(  5.0) F          : (  5.1) (          (  5.2) S          (  5.3) )
(  5.0) F          : (  6.1) V
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<		(0.1)				(1)
	|>		(0.3)		REDUCE	0.0	(3)
	|+		(1.2)				(4)
	|*		(3.2)				(6)
	|(		(5.1)				(8)
	|)		(5.3)		REDUCE	5.0	(16)
	|V		(6.1)		REDUCE	5.0	(20)
3.2	5.0	|	(3.3)	NO SHIFT	REDUCE	3.0	(34)
1.2	5.0	|	(4.1)	NO SHIFT	REDUCE	3.0	(38)
0.1	5.0	|	(4.1)	NO SHIFT	REDUCE	3.0	(39)
5.1	5.0	|	(4.1)	NO SHIFT	REDUCE	3.0	(42)
1.2	3.0	| >	(1.3)	NO SHIFT	REDUCE	1.0	(48)
1.2	3.0	| +	(1.3)	NO SHIFT	REDUCE	1.0	(49)
1.2	3.0	|)	(1.3)	NO SHIFT	REDUCE	1.0	(50)
0.1	3.0	| >	(2.1)	NO SHIFT	REDUCE	1.0	(51)
0.1	3.0	| +	(2.1)	NO SHIFT	REDUCE	1.0	(52)
5.1	3.0	|)	(2.1)	NO SHIFT	REDUCE	1.0	(53)
5.1	3.0	| +	(2.1)	NO SHIFT	REDUCE	1.0	(54)
0.1	3.0	| *	(3.1)	NO SHIFT			(55)
1.2	3.0	| *	(3.1)	NO SHIFT			(56)
5.1	3.0	| *	(3.1)	NO SHIFT			(57)
	1.0	| >	(0.2)	NO SHIFT			(58)
	1.0	| +	(1.1)	NO SHIFT			(59)
	1.0	|)	(5.2)	NO SHIFT			(61)
							*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3
1.0	1.1	1.2	1.3
1.0	2.1		
3.0	3.1	3.2	3.3
3.0	4.1		
5.0	5.1	5.2	5.3
5.0	6.1		

G3 (FEL68) P. 79 TABLE COURTE A ACCES SEQUENTIEL(<=1,<=1)

CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>= 7
LENGTH OF LEFT CONTEXT= 1
LENGTH OF RIGHT CONTEXT= 1

```
                                                        BEGINGRAM
Z   .<. S .>.                                             *
S   S .+. T | T                                           *
T   T .*. F | F                                           *
F   .(. S .). | .V.
                                                        ENDGRAM
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT SEQUENTIAL TABLE

		|<	(0.1)				(1)
		|>	(0.3)		REDUCE	0.0	(3)
		|+	(1.2)				(4)
		|*	(3.2)				(6)
		|(	(5.1)				(8)
		|)	(5.3)		REDUCE	5.0	(16)
		|V	(6.1)		REDUCE	5.0	(20)
3.2	5.0	|	(3.3)	NO SHIFT	REDUCE	3.0	(34)
1.2	5.0	|	(4.1)	NO SHIFT	REDUCE	3.0	(38)
0.1	5.0	|	(4.1)	NO SHIFT	REDUCE	3.0	(39)
5.1	5.0	|	(4.1)	NO SHIFT	REDUCE	3.0	(42)
1.2	3.0	| >	(1.3)	NO SHIFT	REDUCE	1.0	(48)
1.2	3.0	| +	(1.3)	NO SHIFT	REDUCE	1.0	(49)
1.2	3.0	|)	(1.3)	NO SHIFT	REDUCE	1.0	(50)
0.1	3.0	| >	(2.1)	NO SHIFT	REDUCE	1.0	(51)
0.1	3.0	| +	(2.1)	NO SHIFT	REDUCE	1.0	(52)
5.1	3.0	|)	(2.1)	NO SHIFT	REDUCE	1.0	(53)
5.1	3.0	| +	(2.1)	NO SHIFT	REDUCE	1.0	(54)
	3.0	| *	(3.1)	NO SHIFT			(55)
	1.0	| >	(0.2)	NO SHIFT			(58)
	1.0	| +	(1.1)	NO SHIFT			(59)
5.1	1.0	|	(5.2)	NO SHIFT			(61)
							*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3
1.0	1.1	1.2	1.3
1.0	2.1		
3.0	3.1	3.2	3.3
3.0	4.1		
5.0	5.1	5.2	5.3
5.0	6.1		

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z          : (  0.1) <          (  0.2) S          (  0.3) >

(  1.0) S          : (  1.1) S          (  1.2) +          (  1.3) T
(  1.0) S          : (  2.1) T

(  3.0) T          : (  3.1) T          (  3.2) *          (  3.3) F
(  3.0) T          : (  4.1) F

(  5.0) F          : (  5.1) (          (  5.2) S          (  5.3) )
(  5.0) F          : (  6.1) V
```

G4 TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)

```
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1
```

```
                                                                BEGINGRAM
Z   .<. S .>.                                                           *
S   A .X. | B | .X. C | D                                               *
A   .A. A | .A.                                                         *
B   .A. B | .A.                                                         *
C   .C. C | .C.                                                         *
D   .C. D | .C.
                                                                ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z      : (  0.1) <      (  0.2) S      (  0.3) >

(  1.0) S      : (  1.1) A      (  1.2) X
(  1.0) S      : (  2.1) B
(  1.0) S      : (  3.1) X      (  3.2) C
(  1.0) S      : (  4.1) D

(  5.0) A      : (  5.1) A      (  5.2) A
(  5.0) A      : (  6.1) A

(  7.0) B      : (  7.1) A      (  7.2) B
(  7.0) B      : (  8.1) A

(  9.0) C      : (  9.1) C      (  9.2) C
(  9.0) C      : ( 10.1) C

( 11.0) D      : ( 11.1) C      ( 11.2) D
( 11.0) D      : ( 12.1) C
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<		(0.1)				(1)
	|>		(0.3)		REDUCE	0.0	(8)
1.1	|X		(1.2)		REDUCE	1.0	(9)
0.1	|X		(3.1)				(10)
	|A	A	(5.1)				(12)
	|A	X	(6.1)		REDUCE	5.0	(16)
	|A	>	(8.1)		REDUCE	7.0	(23)
3.1	|C	C	(9.1)				(26)
9.1	|C	C	(9.1)				(27)
3.1	|C	>	(10.1)		REDUCE	9.0	(30)
9.1	|C	>	(10.1)		REDUCE	9.0	(31)
0.1	|C	C	(11.1)				(33)
11.1	|C	C	(11.1)				(34)
0.1	|C	>	(12.1)		REDUCE	11.0	(37)
11.1	|C	>	(12.1)		REDUCE	11.0	(38)
0.1	11.0	|	(4.1)	NO SHIFT	REDUCE	1.0	(40)
11.1	11.0	|	(11.2)	NO SHIFT	REDUCE	11.0	(41)
3.1	9.0	|	(3.2)	NO SHIFT	REDUCE	1.0	(43)
9.1	9.0	|	(9.2)	NO SHIFT	REDUCE	9.0	(44)
0.1	7.0	|	(2.1)	NO SHIFT	REDUCE	1.0	(46)
5.1	7.0	|	(7.2)	NO SHIFT	REDUCE	7.0	(47)
0.1	5.0	|	(1.1)	NO SHIFT			(49)
5.1	5.0	|	(5.2)	NO SHIFT	REDUCE	5.0	(50)
	1.0	|	(0.2)	NO SHIFT			(52)
							*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3
1.0	1.1	1.2	
1.0	2.1		
1.0	3.1	3.2	
1.0	4.1		
5.0	5.1	5.2	
5.0	6.1		
7.0	5.1	7.2	
7.0	8.1		
9.0	9.1	9.2	
9.0	10.1		
11.0	11.1	11.2	
11.0	12.1		

```
G5 EXPRESSION ARITMETIQUE ALGOL  TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1



                                                                BEGINGRAM
Z  .<. ASSIGNST .>.                                                     *
ASSIGNST  .ID. .:=. AE                                                  *
AE  TERM | .ADOP. TERM | AE .ADOP. TERM                                 *
TERM  FACTOR | TERM .MULTOP. FACTOR                                     *
FACTOR  PRIMARY | FACTOR .**. PRIMARY                                   *
PRIMARY  .ID. | .(. AE .).
                                                                ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z          : (  0.1) <          (  0.2) ASSIGNST   (  0.3) >

(  1.0) ASSIGNST   : (  1.1) ID         (  1.2) :=         (  1.3) AE

(  2.0) AE         : (  2.1) TERM
(  2.0) AE         : (  3.1) ADOP       (  3.2) TERM
(  2.0) AE         : (  4.1) AE         (  4.2) ADOP       (  4.3) TERM

(  5.0) TERM       : (  5.1) FACTOR
(  5.0) TERM       : (  6.1) TERM       (  6.2) MULTOP     (  6.3) FACTOR

(  7.0) FACTOR     : (  7.1) PRIMARY
(  7.0) FACTOR     : (  8.1) FACTOR     (  8.2) **         (  8.3) PRIMARY

(  9.0) PRIMARY    : (  9.1) ID
(  9.0) PRIMARY    : ( 10.1) (          ( 10.2) AE         ( 10.3) )
```

BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	| <		(0.1)				(1)
	| >		(0.3)		REDUCE	0.0	(2)
0.1	| ID		(1.1)				(3)
1.2	| ID		(9.1)		REDUCE	9.0	(4)
8.2	| ID		(9.1)		REDUCE	9.0	(5)
6.2	| ID		(9.1)		REDUCE	9.0	(6)
3.1	| ID		(9.1)		REDUCE	9.0	(8)
4.2	| ID		(9.1)		REDUCE	9.0	(9)
10.1	| ID		(9.1)		REDUCE	9.0	(12)
	| :=		(1.2)				(32)
1.2	| ADOP		(3.1)				(35)
10.1	| ADOP		(3.1)				(36)
4.1	| ADOP		(4.2)				(39)
	| MULTOP		(6.2)				(41)
	| **		(8.2)				(43)
	| (		(10.1)				(45)
	|)		(10.3)		REDUCE	9.0	(63)
1.2	9.0	|	(7.1)	NO SHIFT	REDUCE	7.0	(68)
6.2	9.0	|	(7.1)	NO SHIFT	REDUCE	7.0	(69)
3.1	9.0	|	(7.1)	NO SHIFT	REDUCE	7.0	(71)
4.2	9.0	|	(7.1)	NO SHIFT	REDUCE	7.0	(72)
10.1	9.0	|	(7.1)	NO SHIFT	REDUCE	7.0	(75)
8.2	9.0	|	(8.3)	NO SHIFT	REDUCE	7.0	(91)
1.2	7.0	| >	(5.1)	NO SHIFT	REDUCE	5.0	(96)
3.1	7.0	| >	(5.1)	NO SHIFT	REDUCE	5.0	(97)
4.2	7.0	| >	(5.1)	NO SHIFT	REDUCE	5.0	(98)
1.2	7.0	| MULTOP	(5.1)	NO SHIFT	REDUCE	5.0	(99)
1.2	7.0	| ADOP	(5.1)	NO SHIFT	REDUCE	5.0	(100)
10.1	7.0	|)	(5.1)	NO SHIFT	REDUCE	5.0	(101)
3.1	7.0	| MULTOP	(5.1)	NO SHIFT	REDUCE	5.0	(102)
4.2	7.0	| MULTOP	(5.1)	NO SHIFT	REDUCE	5.0	(103)
10.1	7.0	| MULTOP	(5.1)	NO SHIFT	REDUCE	5.0	(104)
10.1	7.0	| ADOP	(5.1)	NO SHIFT	REDUCE	5.0	(105)
3.1	7.0	| ADOP	(5.1)	NO SHIFT	REDUCE	5.0	(106)
3.1	7.0	|)	(5.1)	NO SHIFT	REDUCE	5.0	(107)
4.2	7.0	| ADOP	(5.1)	NO SHIFT	REDUCE	5.0	(108)
4.2	7.0	|)	(5.1)	NO SHIFT	REDUCE	5.0	(109)
6.2	7.0	| >	(6.3)	NO SHIFT	REDUCE	5.0	(110)
6.2	7.0	| ADOP	(6.3)	NO SHIFT	REDUCE	5.0	(111)
6.2	7.0	| MULTOP	(6.3)	NO SHIFT	REDUCE	5.0	(112)
6.2	7.0	|)	(6.3)	NO SHIFT	REDUCE	5.0	(113)
1.2	7.0	| **	(8.1)	NO SHIFT			(114)
3.1	7.0	| **	(8.1)	NO SHIFT			(115)
4.2	7.0	| **	(8.1)	NO SHIFT			(116)
6.2	7.0	| **	(8.1)	NO SHIFT			(117)
10.1	7.0	| **	(8.1)	NO SHIFT			(118)
1.2	5.0	| >	(2.1)	NO SHIFT	REDUCE	2.0	(119)

```
 1.2     5.0    | ADOP      (  2.1)   NO SHIFT  REDUCE    2.0           ( 120)
10.1     5.0    | )         (  2.1)   NO SHIFT  REDUCE    2.0           ( 121)
10.1     5.0    | ADOP      (  2.1)   NO SHIFT  REDUCE    2.0           ( 122)
 3.1     5.0    | >         (  3.2)   NO SHIFT  REDUCE    2.0           ( 123)
 3.1     5.0    | ADOP      (  3.2)   NO SHIFT  REDUCE    2.0           ( 124)
 3.1     5.0    | )         (  3.2)   NO SHIFT  REDUCE    2.0           ( 125)
 4.2     5.0    | >         (  4.3)   NO SHIFT  REDUCE    2.0           ( 126)
 4.2     5.0    | ADOP      (  4.3)   NO SHIFT  REDUCE    2.0           ( 127)
 4.2     5.0    | )         (  4.3)   NO SHIFT  REDUCE    2.0           ( 128)
 1.2     5.0    | MULTOP    (  6.1)   NO SHIFT                          ( 129)
 3.1     5.0    | MULTOP    (  6.1)   NO SHIFT                          ( 130)
 4.2     5.0    | MULTOP    (  6.1)   NO SHIFT                          ( 131)
10.1     5.0    | MULTOP    (  6.1)   NO SHIFT                          ( 132)
         2.0    | >         (  1.3)   NO SHIFT  REDUCE    1.0           ( 133)
         2.0    | ADOP      (  4.1)   NO SHIFT                          ( 134)
         2.0    | )         ( 10.2)   NO SHIFT                          ( 136)
         1.0    |           (  0.2)   NO SHIFT                          ( 137)

STRINGS TO BE REDUCED
  0.0      0.1    0.2    0.3
  1.0      1.1    1.2    1.3
  2.0      2.1
  2.0      3.1    3.2
  2.0      4.1    4.2    4.3
  5.0      5.1
  5.0      6.1    6.2    6.3
  7.0      7.1
  7.0      8.1    8.2    8.3
  9.0      9.1
  9.0     10.1   10.2   10.3
```

```
G6 (KNU65) P. 619                    TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1


                                                                        BEGINGRAM
Z   .<. B .>.                                                                   *
B   .A. | L R                                                                   *
L   .A. | L N .B.                                                               *
R   .A. | .B. N R                                                               *
N .A. | .B. N N .B.
                                                                          ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z       : (  0.1) <       (  0.2) B       (  0.3) >

(  1.0) B       : (  1.1) A
(  1.0) B       : (  2.1) L       (  2.2) R

(  3.0) L       : (  3.1) A
(  3.0) L       : (  4.1) L       (  4.2) N       (  4.3) B

(  5.0) R       : (  5.1) A
(  5.0) R       : (  6.1) B       (  6.2) N       (  6.3) R

(  7.0) N       : (  7.1) A
(  7.0) N       : (  8.1) B       (  8.2) N       (  8.3) N       (  8.4) B
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACEL
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<		(0.1)				(1)
	|>		(0.3)		REDUCE	0.0	(3)
0.1	|A	>	(1.1)		REDUCE	1.0	(4)
0.1	|A	A	(3.1)		REDUCE	3.0	(5)
0.1	|A	B	(3.1)		REDUCE	3.0	(7)
2.1	|A	>	(5.1)		REDUCE	5.0	(9)
6.2	|A	>	(5.1)		REDUCE	5.0	(10)
2.1	|A	B	(7.1)		REDUCE	7.0	(11)
6.1	|A	A	(7.1)		REDUCE	7.0	(12)
6.2	|A	B	(7.1)		REDUCE	7.0	(14)
6.1	|A	B	(7.1)		REDUCE	7.0	(15)
4.2	|B		(4.3)		REDUCE	3.0	(17)
2.1	|B		(6.1)				(21)
6.2	|B		(6.1)				(22)
6.1	|B		(6.1)				(26)
8.3	|B		(8.4)		REDUCE	7.0	(33)
2.1	7.0	|	(4.2)	NO SHIFT			(39)
6.1	7.0	|	(6.2)	NO SHIFT			(40)
6.2	7.0	|	(8.3)	NO SHIFT			(44)
2.1	5.0	|	(2.2)	NO SHIFT	REDUCE	1.0	(45)
6.2	5.0	|	(6.3)	NO SHIFT	REDUCE	5.0	(46)
	3.0	|	(2.1)	NO SHIFT			(47)
	1.0	|	(0.2)	NO SHIFT			(51)
							*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3	
1.0	1.1			
1.0	2.1	2.2		
3.0	3.1			
3.0	2.1	4.2	4.3	
5.0	5.1			
5.0	6.1	6.2	6.3	
7.0	7.1			
7.0	6.1	6.2	8.3	8.4

```
G7 (M. BRANQUART,M.B.L.E.)        TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1



                                                              BEGINGRAM
Z   .<. S .>.                                                        *
S   .X. A .A. | .Y. A                                                *
A   .A. A | .A.
                                                              ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z          : (  0.1) <          (  0.2) S          (  0.3) >

(  1.0) S          : (  1.1) X          (  1.2) A          (  1.3) A
(  1.0) S          : (  2.1) Y          (  2.2) A

(  3.0) A          : (  3.1) A          (  3.2) A
(  3.0) A          : (  4.1) A
```

```
STARTER RELATION HAS BEEN COMPUTED
AFTER RELATION HAS BEEN COMPUTED
GENERATION STARTS FOR SYMBOL <          WITH   1* 25=   25 GENERATORS OF    7 WORDS
GENERATION STARTS FOR SYMBOL >          WITH   1* 25=   25 GENERATORS OF    7 WORDS
GENERATION STARTS FOR SYMBOL X          WITH   1* 25=   25 GENERATORS OF    7 WORDS
GENERATION STARTS FOR SYMBOL A          WITH   3* 25=   75 GENERATORS OF    7 WORDS
GENERATION STARTS FOR SYMBOL Y          WITH.  1* 25=   25 GENERATORS OF    7 WORDS
GENERATION STARTS FOR SYMBOL A          WITH   3* 25=   75 GENERATORS OF    7 WORDS
GENERATION STARTS FOR SYMBOL S          WITH   1* 25=   25 GENERATORS OF    7 WORDS
   27 STRINGS HAVE BEEN GENERATED
GRAMMAR NOT COORD FOR   3.1 AND   4.1 STRINGS    7 AND    13
GRAMMAR NOT COORD FOR   3.1 AND   4.1 STRINGS    9 AND    15
GRAMMAR NOT COORD FOR   3.1 AND   4.1 STRINGS    9 AND    18
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK,
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

LENGTHY TABLE

	|<	X	(0.1)				(1)
	|<	Y	(0.1)				(2)
0.2	|>		(0.3)		REDUCE	0.0	(3)
0.1	|X	A	(1.1)				(4)
1.2	|A	>	(1.3)		REDUCE	1.0	(6)
1.1	|A	A	(3.1)				(7)
2.1	|A	A	(3.1)				(8)
3.1	|A	A	(3.1)				(9)
1.1	|A	A	(4.1)		REDUCE	3.0	(13)
2.1	|A	>	(4.1)		REDUCE	3.0	(14)
3.1	|A	A	(4.1)		REDUCE	3.0	(15)
3.1	|A	>	(4.1)		REDUCE	3.0	(16)
0.1	|Y	A	(2.1)				(19)
1.1	3.0 |	A	(1.2)	NO SHIFT			(21)
2.1	3.0 |	>	(2.2)	NO SHIFT	REDUCE	1.0	(22)
3.1	3.0 |	A	(3.2)	NO SHIFT	REDUCE	3.0	(23)
3.1	3.0 |	>	(3.2)	NO SHIFT	REDUCE	3.0	(24)
0.1	1.0 |	>	(0.2)	NO SHIFT			(27)

*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3
1.0	1.1	1.2	1.3
1.0	2.1	2.2	
3.0	3.1	3.2	
3.0	4.1		

G8 (M. GRIFFITHS,I.M.A.G.). TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)

CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>= 7
LENGTH OF LEFT CONTEXT= 1
LENGTH OF RIGHT CONTEXT= 1

```
                                                                  BEGINGRAM
                                                                      *
                                                                      *
Z   .<. IFST .>.
IFST  .IF. .(. EX .). .=. EX .;. | .IF. EX .=. EX .THEN. .ST.
EX   .(. EX .). | .A.                                          ENDGRAM
```

```
INPUT OF THE GRAMMAR COMPLETED

(  0.0) Z          : (  0.1) <          (  0.2) IFST        (  0.3) >

(  1.0) IFST       : (  1.1) IF         (  1.2) (           (  1.3) EX        (  1.4) )          (  1.5) =          (  1.6) EX
(  1.7) ;
(  1.0) IFST       : (  2.1) IF         (  2.2) EX          (  2.3) =          (  2.4) EX         (  2.5) THEN       (  2.6) ST

(  3.0) EX         : (  3.1) (          (  3.2) EX          (  3.3) )
(  3.0) EX         : (  4.1) A
```

```
STARTER RELATION HAS BEEN COMPUTED
AFTER RELATION HAS BEEN COMPUTED
GENERATION STARTS FOR SYMBOL <          WITH   1* 25=    25 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL >          WITH   1* 25=    25 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL IF         WITH   2* 25=    50 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL (          WITH   2* 25=    50 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL )          WITH   2* 25=    50 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL =          WITH   2* 25=    50 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL ;          WITH   1* 25=    25 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL THEN       WITH   1* 25=    25 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL ST         WITH   1* 25=    25 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL A          WITH   1* 25=    25 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL EX         WITH   5* 25=   125 GENERATORS OF     7 WORDS
GENERATION STARTS FOR SYMBOL IFST       WITH   1* 25=    25 GENERATORS OF     7 WORDS
   42 STRINGS HAVE BEEN GENERATED
GRAMMAR NOT COORD FOR   1.4 AND   3.3 STRINGS    19 AND    22
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

LENGTHY TABLE

	|<	IF	(0.1)				(1)
0.2	|>	~	(0.3)		REDUCE	0.0	(3)
0.1	|IF	(	(1.1)				(4)
0.1	|IF	A	(1.1)				(6)
1.1	|(	(	(1.2)				(7)
1.1	|(	A	(1.2)				(8)
1.2	|(	(	(1.2)				(9)
1.5	|(	(	(1.2)				(10)
2.3	|(	(	(1.2)				(12)
1.2	|(	A	(1.2)				(14)
1.5	|(	A	(1.2)				(15)
2.3	|(	A	(1.2)				(17)
1.3	|)	=	(1.4)				(19)
1.3	|)	)	(3.3)		REDUCE	3.0	(20)
1.3	|)	;	(3.3)		REDUCE	3.0	(21)
1.3	|)	=	(3.3)		REDUCE	3.0	(22)
1.3	|)	THEN	(3.3)		REDUCE	3.0	(23)
1.4	|=	(	(1.5)				(25)
1.4	|=	A	(1.5)				(26)
2.2	|=	(	(2.3)				(27)
2.2	|=	A	(2.3)				(28)
1.6	|;	>	(1.7)		REDUCE	1.0	(29)
2.4	|THEN	ST	(2.5)				(30)
2.5	|ST	>	(2.6)		REDUCE	1.0	(31)
1.2	|A	)	(4.1)		REDUCE	3.0	(32)
1.5	|A	;	(4.1)		REDUCE	3.0	(33)
1.1	|A	=	(4.1)		REDUCE	3.0	(34)
2.3	|A	THEN	(4.1)		REDUCE	3.0	(35)
1.2	3.0 |	)	(1.3)	NO SHIFT			(37)
1.5	3.0 |	;	(1.6)	NO SHIFT			(38)
1.1	3.0 |	=	(2.2)	NO SHIFT			(39)
2.3	3.0 |	THEN	(2.4)	NO SHIFT			(40)
0.1	1.0 |	>	(0.2)	NO SHIFT			(42)

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3				
1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7
1.0	1.1	2.2	2.3	2.4	2.5	2.6	
3.0	1.2	1.3	3.3				
3.0	4.1						

G9 (BOR71) P. 94 TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)

CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>= 7
LENGTH OF LEFT CONTEXT= 1
LENGTH OF RIGHT CONTEXT= 1

```
                                                        BEGINGRAM
Z   .<. S .>.                                             *
S   .A. S | .A. T .B. S | .C.                             *
T   .A. T .B. T | .C.
                                                        ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z       : (  0.1) <     (  0.2) S     (  0.3) >

(  1.0) S       : (  1.1) A     (  1.2) S
(  1.0) S       : (  2.1) A     (  2.2) T     (  2.3) B     (  2.4) S
(  1.0) S       : (  3.1) C

(  4.0) T       : (  4.1) A     (  4.2) T     (  4.3) B     (  4.4) T
(  4.0) T       : (  5.1) C
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<		(0.1)				(1)
	|>		(0.3)		REDUCE	0.0	(4)
	|A		(1.1)				(5)
	|B		(2.3)				(26)
	|C	>	(3.1)		REDUCE	1.0	(31)
	|C	B	(5.1)		REDUCE	4.0	(36)
1.1	4.0	|	(2.2)	NO SHIFT			(40)
2.3	4.0	|	(4.4)	NO SHIFT	REDUCE	4.0	(42)
0.1	1.0	|	(0.2)	NO SHIFT			(44)
1.1	1.0	|	(1.2)	NO SHIFT	REDUCE	1.0	(45)
2.3	1.0	|	(2.4)	NO SHIFT	REDUCE	1.0	(48)
							*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3	
1.0	1.1	1.2		
1.0	1.1	2.2	2.3	2.4
1.0	3.1			
4.0	1.1	2.2	2.3	4.4
4.0	5.1			

G11 TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)

```
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>= 25
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1
```

```
                                                                BEGINGRAM
                                                                       *
Z   .<. S .>.                                                          *
S   .A. A .B. S | .B. B .A. S | .NOSTRING.                             *
A   .A. A .B. A | .NOSTRING.
B   .B. B .A. B | .NOSTRING.
                                                                ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z      : (  0.1) <      (  0.2) S      (  0.3) >

(  1.0) S      : (  1.1) A      (  1.2) A      (  1.3) B      (  1.4) S
(  1.0) S      : (  2.1) B      (  2.2) B      (  2.3) A      (  2.4) S
(  1.0) S      : (  3.1) NOSTRING

(  4.0) A      : (  4.1) A      (  4.2) A      (  4.3) B      (  4.4) A
(  4.0) A      : (  5.1) NOSTRING

(  6.0) B      : (  6.1) B      (  6.2) B      (  6.3) A      (  6.4) B
(  6.0) B      : (  7.1) NOSTRING
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.O WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.O:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<		(0.1)			(1)
0.2	|>		(0.3)		REDUCE 0.0	(4)
0.1	|A		(1.1)			(5)
1.3	|A		(1.1)			(6)
2.3	|A		(1.1)			(7)
2.2	|A		(2.3)			(11)
1.1	|A		(4.1)			(14)
4.1	|A		(4.1)			(15)
4.3	|A		(4.1)			(16)
6.2	|A		(6.3)			(20)
1.2	|B		(1.3)			(23)
0.1	|B		(2.1)			(26)
1.3	|B		(2.1)			(27)
2.3	|B		(2.1)			(28)
4.2	|B		(4.3)			(32)
2.1	|B		(6.1)			(35)
6.1	|B		(6.1)			(36)
6.3	|B		(6.1)			(37)
0.1	|>		(3.1)	NO SHIFT	REDUCE 1.0	(41)
1.3	|>		(3.1)	NO SHIFT	REDUCE 1.0	(42)
2.3	|>		(3.1)	NO SHIFT	REDUCE 1.0	(43)
1.1	|B		(5.1)	NO SHIFT	REDUCE 4.0	(44)
4.1	|B		(5.1)	NO SHIFT	REDUCE 4.0	(45)
4.3	|B		(5.1)	NO SHIFT	REDUCE 4.0	(46)
2.1	|A		(7.1)	NO SHIFT	REDUCE 6.0	(60)
6.1	|A		(7.1)	NO SHIFT	REDUCE 6.0	(61)
6.3	|A		(7.1)	NO SHIFT	REDUCE 6.0	(62)
2.1	6.0	|	(2.2)	NO SHIFT		(70)
6.1	6.0	|	(6.2)	NO SHIFT		(71)
6.3	6.0	|	(6.4)	NO SHIFT	REDUCE 6.0	(72)
1.1	4.0	|	(1.2)	NO SHIFT		(80)
4.1	4.0	|	(4.2)	NO SHIFT		(81)
4.3	4.0	|	(4.4)	NO SHIFT	REDUCE 4.0	(82)
0.1	1.0	|	(0.2)	NO SHIFT		(84)
1.3	1.0	|	(1.4)	NO SHIFT	REDUCE 1.0	(85)
2.3	1.0	|	(2.4)	NO SHIFT	REDUCE 1.0	(86)

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3	
1.0	1.1	1.2	1.3	1.4
1.0	2.1	2.2	2.3	2.4
1.0	3.1	4.2	4.3	4.4
4.0	4.1			
4.0	5.1	6.2	6.3	6.4
6.0	6.1			
6.0	7.1			

```
G12 APL (M. KALFON,I.R.I.A.)      TABLE COURTE A ACCES ALEATOIRE(<=1,<=1)
CARD(V)<=100
CARD(N)<= 50
CARD(T)<= 50
LENGTH OF ALL THE PRODUCTIONS<=300
NUMBER OF GENERATORS PER SYMBOL>=185
LENGTH OF EACH GENERATOR>=  7
LENGTH OF LEFT CONTEXT=  1
LENGTH OF RIGHT CONTEXT=  1



                                                                    BEGINGRAM
                                                                            *
Z .<. ST .>.
ST   ST .DIAD. .ID. | ST .DIAD. T1 | ST .DIAD. .CONSVEC. | ST .DIAD. .QUAD. |
     ST .DIAD. PRIM | ST .MONA. | ST .DIMO. .ID. | ST .ID. T1 | ST .ID. .CONSVEC.
     | ST .ID. .QUAD. | ST .ID. PRIM | ST .ID. | .ID. | T1 | .CONSVEC. | .QUAD. |
                                                                            *
     PRIM
T1   TSILS .BRA. .CONSVEC. | TSILS .BRA. .QUAD. | TSILS .BRA. PRIM | TSILS .BRA.
                                                                            *
     .ID.                                                                   *
PRIM  .). ST .(.
TSILS  .KET. | .KET. ST | TSILS .;. ST | TSILS .;.
                                                                    ENDGRAM
```

INPUT OF THE GRAMMAR COMPLETED

```
(  0.0) Z        : (  0.1) <      (  0.2) ST      (  0.3) >

(  1.0) ST       : (  1.1) ST     (  1.2) DIAD    (  1.3) ID
(  1.0) ST       : (  2.1) ST     (  2.2) DIAD    (  2.3) T1
(  1.0) ST       : (  3.1) ST     (  3.2) DIAD    (  3.3) CONSVEC
(  1.0) ST       : (  4.1) ST     (  4.2) DIAD    (  4.3) QUAD
(  1.0) ST       : (  5.1) ST     (  5.2) DIAD    (  5.3) PRIM
(  1.0) ST       : (  6.1) ST     (  6.2) MONA
(  1.0) ST       : (  7.1) ST     (  7.2) DIMO    (  7.3) ID
(  1.0) ST       : (  8.1) ST     (  8.2) ID      (  8.3) T1
(  1.0) ST       : (  9.1) ST     (  9.2) ID      (  9.3) CONSVEC
(  1.0) ST       : ( 10.1) ST     ( 10.2) ID      ( 10.3) QUAD
(  1.0) ST       : ( 11.1) ST     ( 11.2) ID      ( 11.3) PRIM
(  1.0) ST       : ( 12.1) ST     ( 12.2) ID
(  1.0) ST       : ( 13.1) ID
(  1.0) ST       : ( 14.1) T1
(  1.0) ST       : ( 15.1) CONSVEC
(  1.0) ST       : ( 16.1) QUAD
(  1.0) ST       : ( 17.1) PRIM

( 18.0) T1       : ( 18.1) TSILS  ( 18.2) BRA     ( 18.3) CONSVEC
( 18.0) T1       : ( 19.1) TSILS  ( 19.2) BRA     ( 19.3) QUAD
( 18.0) T1       : ( 20.1) TSILS  ( 20.2) BRA     ( 20.3) PRIM
( 18.0) T1       : ( 21.1) TSILS  ( 21.2) BRA     ( 21.3) ID

( 22.0) PRIM     : ( 22.1) )      ( 22.2) ST      ( 22.3) (

( 23.0) TSILS    : ( 23.1) KET
( 23.0) TSILS    : ( 24.1) KET    ( 24.2) ST
( 23.0) TSILS    : ( 25.1) TSILS  ( 25.2) ;       ( 25.3) ST
( 23.0) TSILS    : ( 26.1) TSILS  ( 26.2) ;
```

LEFT HANDSIDE OF | REPRESENTS STACK CONTENTS;
RIGHT HANDSIDE OF | REPRESENTS INPUT TAPE CONTENTS;
BETWEEN BRACKETS,SYMBOL TO PUSH INTO THE STACK;
POSSIBLY NOSHIFT IF INPUT TAPE MUST NOT MOVE BY ONE
SYMBOL AS USUAL;POSSIBLY REDUCE I.0 WHEN A COMPLETE
PRODUCTION IS ON TOP OF THE STACK AND MUST BE REPLACED
BY I.0:WHEN USING A SHORT TABLE,CORRECTNESS OF A
PRODUCTION SHOULD BE CHECKED AGAINST THE LIST GIVEN
UNDER TITLE:STRINGS TO BE REDUCED.

SHORT RANDOM TABLE

	|<		(0.1)			(1)
	|>		(0.3)	REDUCE	0.0	(7)
	|DIAD	ID	(1.2)			(8)
	|DIAD	KET	(2.2)			(9)
	|DIAD	CONSVEC	(3.2)			(11)
	|DIAD	QUAD	(4.2)			(12)
	|DIAD	)	(5.2)			(13)
1.2	|ID	>	(1.3)	REDUCE	1.0	(14)
1.2	|ID	DIAD	(1.3)	REDUCE	1.0	(15)
1.2	|ID	MONA	(1.3)	REDUCE	1.0	(20)
1.2	|ID	DIMO	(1.3)	REDUCE	1.0	(21)
1.2	|ID	ID	(1.3)	REDUCE	1.0	(22)
1.2	|ID	BRA	(1.3)	REDUCE	1.0	(27)
1.2	|ID	(	(1.3)	REDUCE	1.0	(31)
1.2	|ID	;	(1.3)	REDUCE	1.0	(32)
7.2	|ID	>	(7.3)	REDUCE	1.0	(34)
7.2	|ID	DIAD	(7.3)	REDUCE	1.0	(35)
7.2	|ID	MONA	(7.3)	REDUCE	1.0	(40)
7.2	|ID	DIMO	(7.3)	REDUCE	1.0	(41)
7.2	|ID	ID	(7.3)	REDUCE	1.0	(42)
7.2	|ID	BRA	(7.3)	REDUCE	1.0	(47)
7.2	|ID	(	(7.3)	REDUCE	1.0	(51)
7.2	|ID	;	(7.3)	REDUCE	1.0	(52)
8.1	|ID	KET	(8.2)			(54)
8.1	|ID	CONSVEC	(9.2)			(56)
8.1	|ID	QUAD	(10.2)			(57)
8.1	|ID	)	(11.2)			(58)
8.1	|ID	>	(12.2)	REDUCE	1.0	(59)
8.1	|ID	DIAD	(12.2)	REDUCE	1.0	(60)
8.1	|ID	MONA	(12.2)	REDUCE	1.0	(65)
8.1	|ID	DIMO	(12.2)	REDUCE	1.0	(66)
8.1	|ID	ID	(12.2)	REDUCE	1.0	(67)
8.1	|ID	(	(12.2)	REDUCE	1.0	(72)
8.1	|ID	BRA	(12.2)	REDUCE	1.0	(73)
8.1	|ID	;	(12.2)	REDUCE	1.0	(78)
0.1	|ID	>	(13.1)	REDUCE	1.0	(85)
0.1	|ID	DIAD	(13.1)	REDUCE	1.0	(86)
0.1	|ID	MONA	(13.1)	REDUCE	1.0	(91)
0.1	|ID	DIMO	(13.1)	REDUCE	1.0	(92)
0.1	|ID	ID	(13.1)	REDUCE	1.0	(93)
22.1	|ID	(	(13.1)	REDUCE	1.0	(98)
24.1	|ID	BRA	(13.1)	REDUCE	1.0	(99)
25.2	|ID	BRA	(13.1)	REDUCE	1.0	(100)

22.1	ID	DIAD	(13.1)	REDUCE	1.0	(101)
24.1	ID	DIAD	(13.1)	REDUCE	1.0	(102)
25.2	ID	DIAD	(13.1)	REDUCE	1.0	(103)
22.1	ID	MONA	(13.1)	REDUCE	1.0	(116)
24.1	ID	MONA	(13.1)	REDUCE	1.0	(117)
25.2	ID	MONA	(13.1)	REDUCE	1.0	(118)
22.1	ID	DIMO	(13.1)	REDUCE	1.0	(119)
24.1	ID	DIMO	(13.1)	REDUCE	1.0	(120)
25.2	ID	DIMO	(13.1)	REDUCE	1.0	(121)
22.1	ID	ID	(13.1)	REDUCE	1.0	(122)
24.1	ID	ID	(13.1)	REDUCE	1.0	(123)
25.2	ID	ID	(13.1)	REDUCE	1.0	(124)
24.1	ID	;	(13.1)	REDUCE	1.0	(140)
25.2	ID	;	(13.1)	REDUCE	1.0	(145)
21.2	ID	>	(21.3)	REDUCE	18.0	(147)
21.2	ID	DIAD	(21.3)	REDUCE	18.0	(148)
21.2	ID	MONA	(21.3)	REDUCE	18.0	(153)
21.2	ID	DIMO	(21.3)	REDUCE	18.0	(154)
21.2	ID	ID	(21.3)	REDUCE	18.0	(155)
21.2	ID	BRA	(21.3)	REDUCE	18.0	(160)
21.2	ID	(	(21.3)	REDUCE	18.0	(164)
21.2	ID	;	(21.3)	REDUCE	18.0	(165)
3.2	CONSVEC		(3.3)	REDUCE	1.0	(167)
9.2	CONSVEC		(9.3)	REDUCE	1.0	(187)
0.1	CONSVEC		(15.1)	REDUCE	1.0	(207)
22.1	CONSVEC		(15.1)	REDUCE	1.0	(220)
24.1	CONSVEC		(15.1)	REDUCE	1.0	(221)
25.2	CONSVEC		(15.1)	REDUCE	1.0	(222)
18.2	CONSVEC		(18.3)	REDUCE	18.0	(269)
4.2	QUAD		(4.3)	REDUCE	1.0	(289)
10.2	QUAD		(10.3)	REDUCE	1.0	(309)
0.1	QUAD		(16.1)	REDUCE	1.0	(329)
22.1	QUAD		(16.1)	REDUCE	1.0	(342)
24.1	QUAD		(16.1)	REDUCE	1.0	(343)
25.2	QUAD		(16.1)	REDUCE	1.0	(344)
19.2	QUAD		(19.3)	REDUCE	18.0	(391)
	MONA		(6.2)	REDUCE	1.0	(411)
	DIMO		(7.2)			(437)
	BRA	CONSVEC	(18.2)			(438)
	BRA	QUAD	(19.2)			(439)
	BRA	)	(20.2)			(440)
	BRA	ID	(21.2)			(441)
	)		(22.1)			(442)
	(		(22.3)	REDUCE	22.0	(484)
	KET	BRA	(23.1)	REDUCE	23.0	(504)
	KET	;	(23.1)	REDUCE	23.0	(508)
	KET	ID	(24.1)			(540)
	KET	CONSVEC	(24.1)			(546)
	KET	QUAD	(24.1)			(547)
	KET	)	(24.1)			(548)
	KET	KET	(24.1)			(549)
	;	ID	(25.2)			(576)
	:	CONSVEC	(25.2)			(577)
	;	QUAD	(25.2)			(578)
	;	)	(25.2)			(579)
	;	KET	(25.2)			(580)
	;	BRA	(26.2)	REDUCE	23.0	(582)
	;	;	(26.2)	REDUCE	23.0	(586)
23.0	BRA		(18.1)	NO SHIFT		(588)
23.0	;		(25.1)	NO SHIFT		(612)

5.2	22.0	|	(5.3)	NO SHIFT	REDUCE	1.0	(624)
11.2	22.0	|	(11.3)	NO SHIFT	REDUCE	1.0	(644)
0.1	22.0	|	(17.1)	NO SHIFT	REDUCE	1.0	(664)
22.1	22.0	|	(17.1)	NO SHIFT	REDUCE	1.0	(677)

24.1	22.0	|	(17.1)	NO SHIFT	REDUCE	1.0	(678)
25.2	22.0	|	(17.1)	NO SHIFT	REDUCE	1.0	(679)
20.2	22.0	|	(20.3)	NO SHIFT	REDUCE	18.0	(726)
2.2	18.0	|	(2.3)	NO SHIFT	REDUCE	1.0	(746)
8.2	18.0	|	(8.3)	NO SHIFT	REDUCE	1.0	(766)
0.1	18.0	|	(14.1)	NO SHIFT	REDUCE	1.0	(786)
22.1	18.0	|	(14.1)	NO SHIFT	REDUCE	1.0	(799)
24.1	18.0	|	(14.1)	NO SHIFT	REDUCE	1.0	(800)
25.2	18.0	|	(14.1)	NO SHIFT	REDUCE	1.0	(801)
0.1	1.0	| >	(0.2)	NO SHIFT			(848)
0.1	1.0	| DIAD	(1.1)	NO SHIFT			(849)
22.1	1.0	| DIAD	(1.1)	NO SHIFT			(850)
24.1	1.0	| DIAD	(1.1)	NO SHIFT			(851)
25.2	1.0	| DIAD	(1.1)	NO SHIFT			(852)
0.1	1.0	| MONA	(6.1)	NO SHIFT			(869)
22.1	1.0	| MONA	(6.1)	NO SHIFT			(870)
24.1	1.0	| MONA	(6.1)	NO SHIFT			(871)
25.2	1.0	| MONA	(6.1)	NO SHIFT			(872)
0.1	1.0	| DIMO	(7.1)	NO SHIFT			(873)
22.1	1.0	| DIMO	(7.1)	NO SHIFT			(874)
24.1	1.0	| DIMO	(7.1)	NO SHIFT			(875)
25.2	1.0	| DIMO	(7.1)	NO SHIFT			(876)
0.1	1.0	| ID	(8.1)	NO SHIFT			(877)
22.1	1.0	| ID	(8.1)	NO SHIFT			(878)
24.1	1.0	| ID	(8.1)	NO SHIFT			(879)
25.2	1.0	| ID	(8.1)	NO SHIFT			(880)
22.1	1.0	| (	(22.2)	NO SHIFT			(897)
24.1	1.0	| BRA	(24.2)	NO SHIFT	REDUCE	23.0	(898)
24.1	1.0	| ;	(24.2)	NO SHIFT	REDUCE	23.0	(902)
25.2	1.0	| BRA	(25.3)	NO SHIFT	REDUCE	23.0	(904)
25.2	1.0	| ;	(25.3)	NO SHIFT	REDUCE	23.0	(908)
							*

STRINGS TO BE REDUCED

0.0	0.1	0.2	0.3
1.0	1.1	1.2	1.3
1.0	1.1	2.2	2.3
1.0	1.1	3.2	3.3
1.0	1.1	4.2	4.3
1.0	1.1	5.2	5.3
1.0	6.1	6.2	
1.0	7.1	7.2	7.3
1.0	8.1	8.2	8.3
1.0	8.1	9.2	9.3
1.0	8.1	10.2	10.3
1.0	8.1	11.2	11.3
1.0	8.1	12.2	
1.0	13.1		
1.0	14.1		
1.0	15.1		
1.0	16.1		
1.0	17.1		
18.0	18.1	18.2	18.3
18.0	18.1	19.2	19.3
18.0	18.1	20.2	20.3
18.0	18.1	21.2	21.3

22.0	22.1	22.2	22.3
23.0	23.1		
23.0	24.1	24.2	
23.0	25.1	25.2	25.3
23.0	25.1	26.2	